COLLEGE OF ENGINEERING, GUINDY

ANNA UNIVERSITY

CHENNAI 600025

EC5612 WIRELESS COMMUNICATION AND NETWORKING LABORATORY (R2019)

RECORD

SUBMITTED BY

Vasanth Kumar V
(2019105064)

In partial fulfilment for the award of the degree of
BACHELOR OF ENGINEERING
IN
ELECTRONICS AND COMMUNICATION ENGINEERING

ANNA UNIVERSITY

COLLEGE OF ENGINEERING GUINDY

BONAFIDE CERTIFICATE

NAME: Vasanth Kumar V

DEPT: B.E. ECE

REGISTER NUMBER: 2019105064

It is certified that this is the bonafide record of the work done by the above-mentioned student in the **WIRELESS COMMUNICATION AND NETWORKING LAB - EC5612** (2019-reg) during the period December 2021 - April 2022.

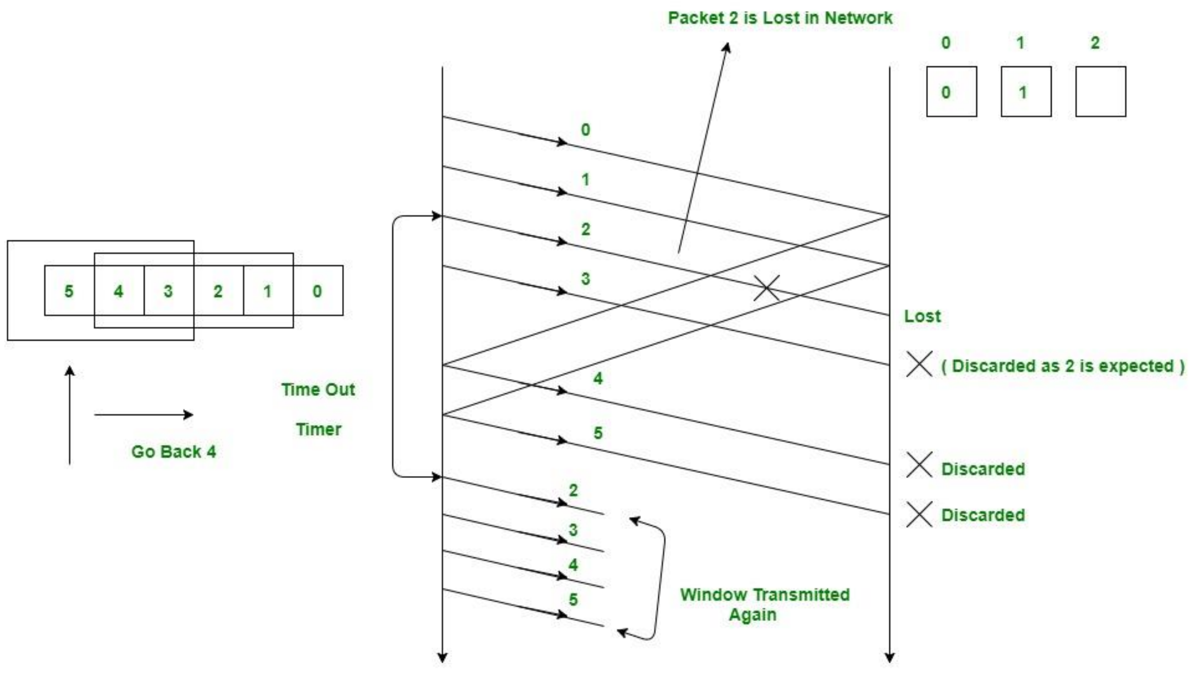
Signature of Lab-In-Charge *Signature of the Head of the Department*

Submitted for the practical exam held on ___________

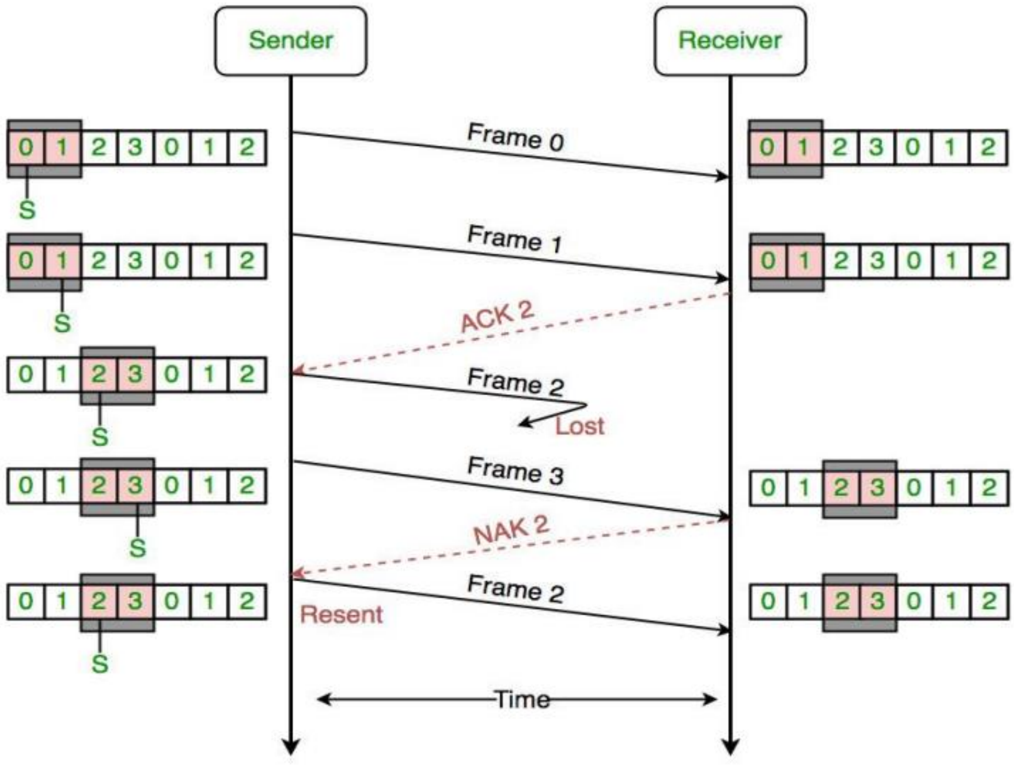
Signature of the Internal Examiner

INDEX

GO BACK N protocol:



SELECTIVE REPEAT protocol:



EXP.NO: 1 DATE: 21.03.2022	LLC PROTOCOLS

AIM:

To implement
- Go Back 'N' Protocol
- Selective Repeat Protocol

and to determine the throughput and delay for each.

SOFTWARE REQUIRED:

NETSIM Software

THEORY:

1) GO BACK 'N' PROTOCOL:

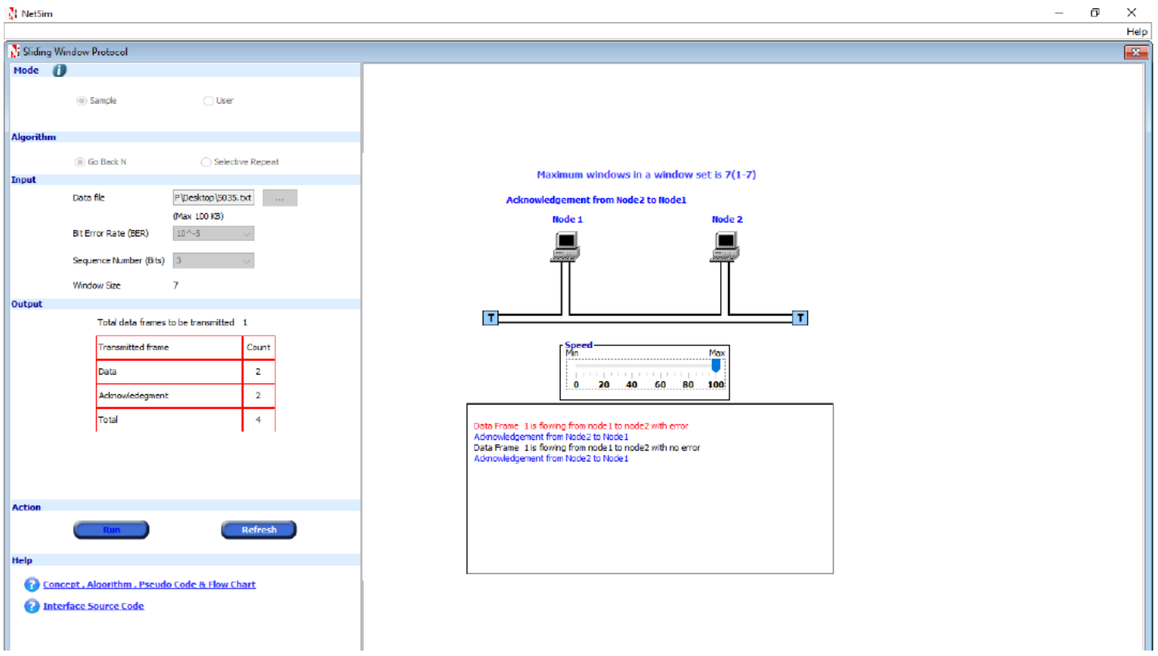
Go Back 'N' is a connection-oriented transmission. The sender transmits the frames continuously. Each frame in the buffer has a sequence number starting from 1 and increasing up to the window size. The sender has a window i.e., a buffer to store the frames. This buffer size is the number of frames to be transmitted continuously. The size of the window depends on the protocol designer.

2) SELECTIVE REPEAT PROTOCOL:

It is similar to Go Back 'N' Protocol, but the sender send frame only after the reception of ACK signal. It may be used as a protocol for delivery and ACK for message units for delivery of subdivided message. It is used as a protocol for delivery of message sender continuous to send frames specifies by windows size even after becoming frameless. Once the sender has sent all the frame in its windows, it resends the frame number given by ACK and continuous where it left off.

LAYOUT

GO BACK 'N' PROTOCOL:



Mode: [✓ Sample] User

Algorithm: [✓ Go Back N] Selective Repeat

Data file	P:\Desktop\5035.txt (Max 100 KB)
Bit Error Rate (BER)	10^-5
Sequence Number (Bits)	3
Window Size	7

Total data frames to be transmitted 1

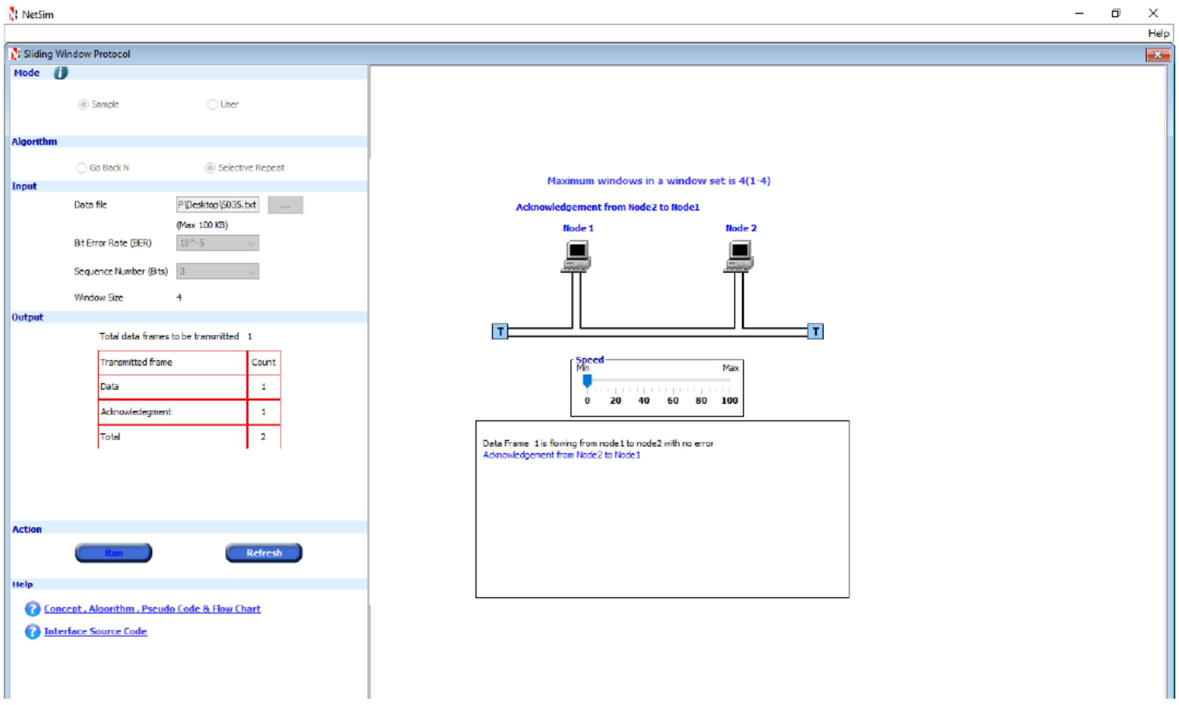
Transmitted frame	Count
Data	2
Acknowledgement	2
Total	4

Maximum windows in a window set is 7(1-7)

Acknowledgement from Node2 to Node1

```
Data Frame  1 is flowing from node1 to node2 with error
Acknowledgement from Node2 to Node1
Data Frame  1 is flowing from node1 to node2 with no error
Acknowledgement from Node2 to Node1
```

SELECTIVE REPEAT PROTOCOL:



Mode: [✓ Sample] User

Algorithm: Go Back N [✓ Selective Repeat]

Data file	P:\Desktop\5035.txt (Max 100 KB)
Bit Error Rate (BER)	10^-5
Sequence Number (Bits)	3
Window Size	4

Total data frames to be transmitted 1

Transmitted frame	Count
Data	1
Acknowledgement	1
Total	2

Maximum windows in a window set is 4(1-4)

Acknowledgement from Node2 to Node1

```
Data Frame  1 is flowing from node1 to node2 with no error
Acknowledgement from Node2 to Node1
```

ALGORITHM:

1) GO BACK 'N' PROTOCOL:

- The source code transmits the frames continuously.
- Each frame in the buffer has a sequence number starting from 1 and increasing up to the window size.
- The source code has a window i.e., a buffer to store the frames. This buffer size is the number of frames to be transmitted continuously.
- The size of the window depends on the protocol designer.
- For the first frame, the receiving node forms a positive acknowledgement if the frame is received without any error.
- If subsequent frames are received without error (up to window size) cumulative positive acknowledgement is formed.
- If the subsequent frame is received with error, the cumulative acknowledgement error-free frames are transmitted. If, In the same window two frames or more frames are received with error, the second and the subsequent error frames are neglected. Similarly, even the frames received without error after the receipt of a frame with error are neglected.
- The source code re-transmits all frames of window from the first error frame.
- If the frames are errorless in the next transmission and if the acknowledgement is error-free, the window slides by the number of error-free frames being transmitted.
- If the acknowledgement is transmitted with error, all the frames of window at source are re-transmitted and window doesn't slide.
- This concept of replacing the transmission from the first error frame in the window is called as **Go Back-N** transmission flow control protocol.

TABULATION:

1. GO BACK N PROTOCOL(Sequence number:3 bits)

BER	TOTAL DATA FRAMES TO BE TRANSMITTED	DATA	ACKNOWLEDGEMENT	TOTAL
10^-5	1	2	2	4
10^-6	1	2	2	4
10^-7	1	1	1	2
10^-8	1	1	1	2
10^-9	1	1	1	2
NO ERROR	1	1	1	2

2. SELECTIVE REPEAT PROTOCOL(Sequence number:3 bits)

BER	TOTAL DATA FRAMES TO BE TRANSMITTED	DATA	ACKNOWLEDGEMENT	TOTAL
10^-5	1	1	1	2
10^-6	1	1	1	2
10^-7	1	1	1	2
10^-8	1	1	1	2
10^-9	1	1	1	2
NO ERROR	1	1	1	2

2) SELECTIVE REPEAT PROTOCOL:

- The source node transmits the frames continuously.
- Each frame in the buffer has a sequence number starting from 1 and increasing up to the window size.
- The source node has a window i.e., buffer to store the frames. This buffer size is the number of frames to be transmitted continuously.
- The receiver has a buffer to store the received frames. The size of the buffer depends upon the window size defined by the protocol designer.
- The source node transmits frames continuously till the window size is exhausted. If any of the frames are received with error only those frames are requested for retransmission (with a negative acknowledgement).
- If all the frames are received without error, a cumulative positive acknowledgement is sent.
- If there is an error in frame 3, an acknowledgement for the frame 2 is sent and then only frame 3 is retransmitted. Now the window slides to get the next frames to the window.
- If acknowledgement is transmitted with error, all the frames of window are retransmitted. Else ordinary window sliding takes place. (*In implementation part, Acknowledgement error is not considered).
- If all the frames transmitted are errorless the next transmission is carried out for the new window.
- This concept of repeating the transmission for the error frames only is called **Selective Repeat** transmission flow control protocol.

PROCEDURE:

1. Open NETSIM.
2. Click Programming >> Transmission Flow Control.
3. Select Sample.
4. Select **Go Back-N** transmission.
5. Enter input data and BER.
6. Click **Link** to execute the program.
7. Repeat steps 1 to 3.
8. Select **Selective Repeat** protocol.
9. Enter input data and BER. Click **Link** to execute the program.

RESULT:

Thus, the LLC protocols such as "Go Back 'N'" and "Selective Repeat" were studied using NETSIM and their performance were analysed.

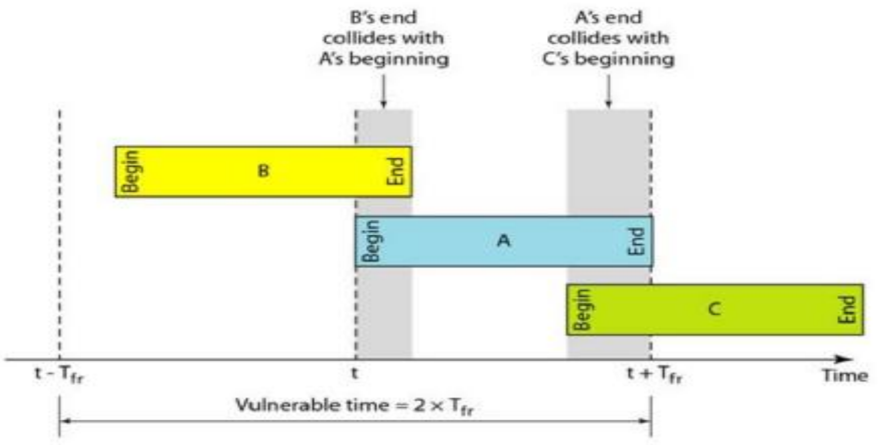
INFERENCE:

It is found that Selective Repeat Protocol is the most optimum out of these two LLC protocols. But the complexity equipment's are high. So, a trade off exists the total number of transmissions taken place and complexity.

CONCLUSION:

Both the LLC protocols were studied and it can be concluded that Selective Repeat is more effective but also more complex as compared to Go back n protocol.

1. PURE ALOHA:



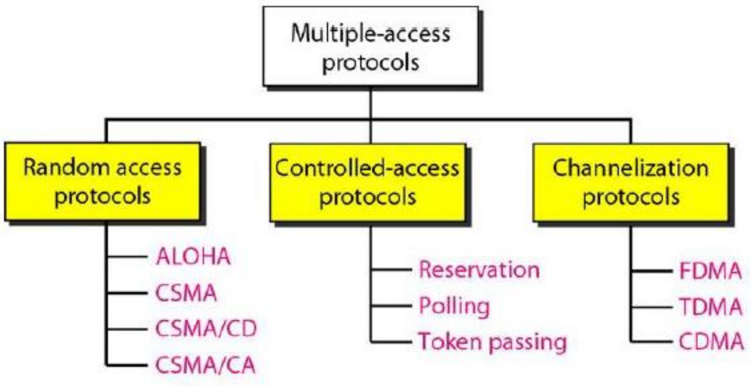
B's end collides with A's beginning
A's end collides with C's beginning
Begin
B
End
Begin
A
End
Begin
C
End
t - T_{fr}
t
t + T_{fr}
Time
Vulnerable time = 2 × T_{fr}

EXP.NO: 2 DATE: 21.3.2022	MAC PROTOCOLS

AIM:

To study and analyse the following MAC protocols:

- Slotted ALOHA
- Pure ALOHA



- CSMA/CA

SOFTWARE REQUIRED:

NETSIM

THEORY:

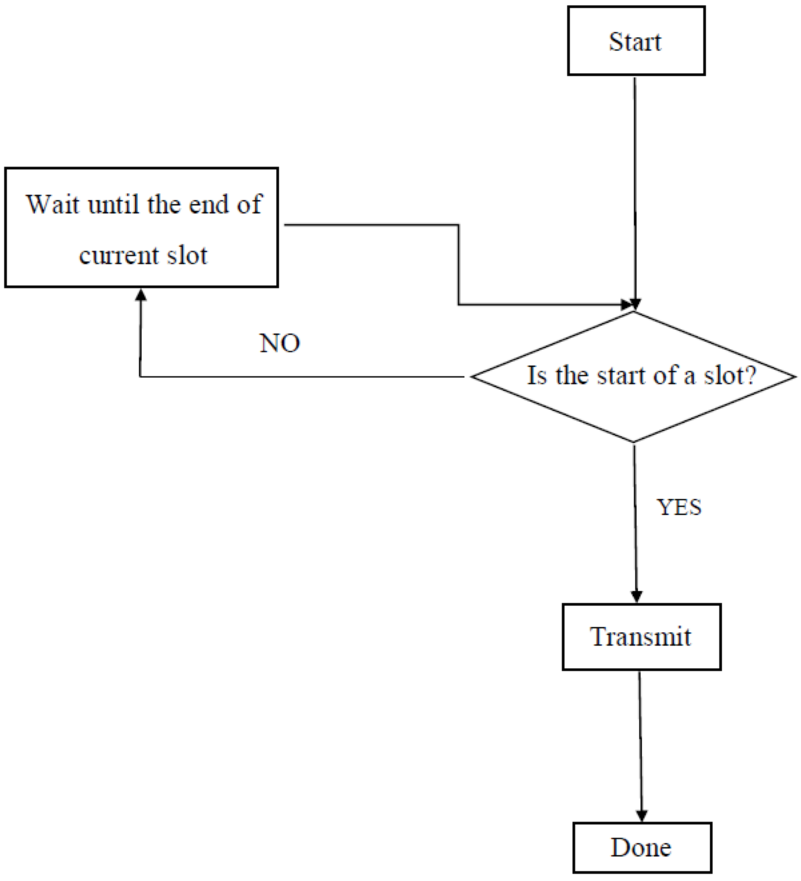
MAC PROTOCOLS:

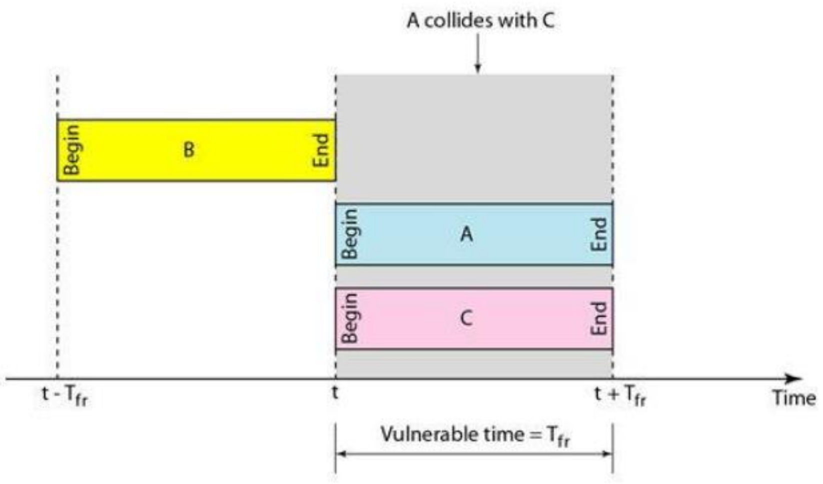
The medium access control (MAC) is a sublayer of the data link layer (DLL) in the seven-layer of the open system interconnections (OSI) reference model for data transmission. MAC is responsible for flow control and multiplexing for transmission medium. It controls the transmission of data packets via remotely shared channels. It sends data over the network interface card.

RANDOM ACCESS PROTOCOL:

In random access or contention methods, no station is superior to another station and none is assigned the control over another. No

2. SLOTTED ALOHA:





station permits, or does not permit, another station to send. At each instance, a station that has data to send uses a procedure defined by the protocol make a decision on whether or not to send.

1. **PURE ALOHA:**

This allows every station to transmit the data whenever they have data to send. If the frame is damaged, the stations wait for a random amount of time and re-transmit the frame till it is transmitted successfully.
The vulnerable time is defined as the time in which the collision may occur.
The throughput of pure ALOHA is S=Ge-2G , which is maximum for G=1/2(18% of the total transmitted frames).

2. **SLOTTED ALOHA:**

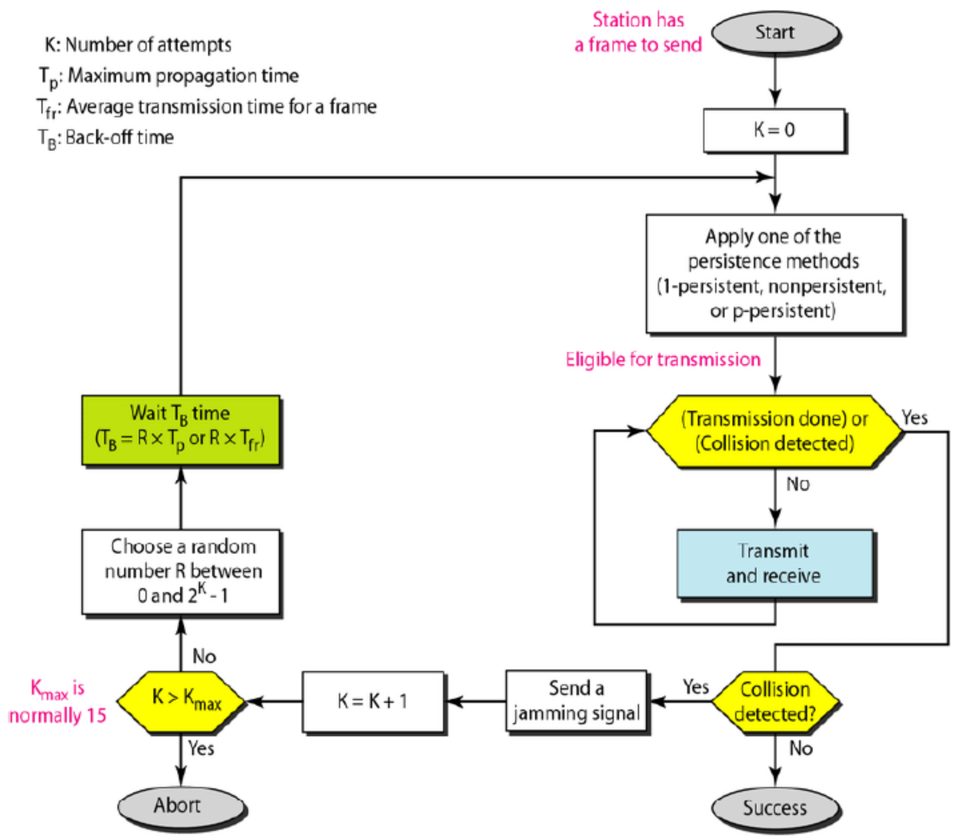
This allows station to transmit data only at specific time slots. So, station cannot transmit whenever it wants,
In this, the vulnerable time is reduced by half.
The throughput of the slotted ALOHA is S=Ge-G which is maximum when G = 1(37%).

3. **CSMA/ CD:**

Carrier Sense Multiple Access (CSMA) with collision detection (CD) is a Medium Access Control Method, used notably in Ethernet Technology. CSMA/CD is used to improve the performance of pure CSMA by terminating transition as soon as a collision is detected. Thus, shortening the time required before a retry can be attempted.

COLLISION DETECTION:
When more than one signal is sent in the cable, the power level exceeds a certain limit which indicates that collision is occurred.

3. CSMA/ CD



K: Number of attempts
T_p: Maximum propagation time
T_{fr}: Average transmission time for a frame
T_B: Back-off time
Station has a frame to send
Start
K = 0
Apply one of the persistence methods (1-persistent, nonpersistent, or p-persistent)
Eligible for transmission
(Transmission done) or (Collision detected)
Yes
No
Transmit and receive
Collision detected?
Yes
No
Success
Send a jamming signal
K = K + 1
$K > K_{max}$
K_{max} is normally 15
No
Yes
Abort
Choose a random number R between 0 and $2^K - 1$
Wait T_B time ($T_B = R \times T_p$ or $R \times T_{fr}$)

MECHANISM:

1. Check if the sender is ready for transmitting data packets

2. Sender has to keep on checking if the transmission link/medium is idle. If it senses that carrier is free and there are no collisions, it sends the data. Otherwise, it retains from sending data.

3. Transmit the data and check for collisions. If collisions is detected, the transmission is stopped and after some transfer data and repeats above process.

4. If no collision was detected, the counter is reset.

PROCEDURE:

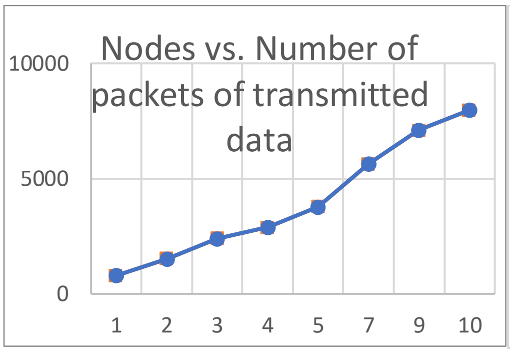
1. PURE ALOHA:

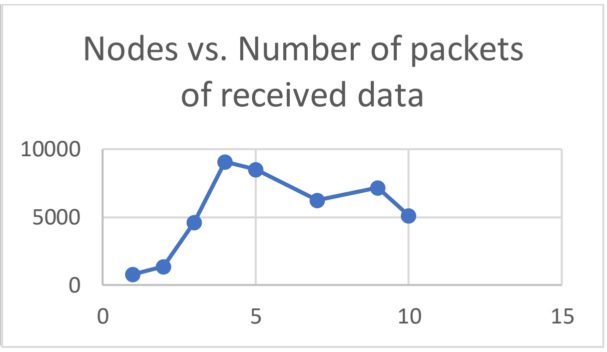
- Select New ->Legacy Networks -> pure ALOHA
- Click and drop required number of nodes.
- Right click on the grid environment and select the channel characteristics as no path loss.
- Click and drop the appropriate icon and set the following properties as,

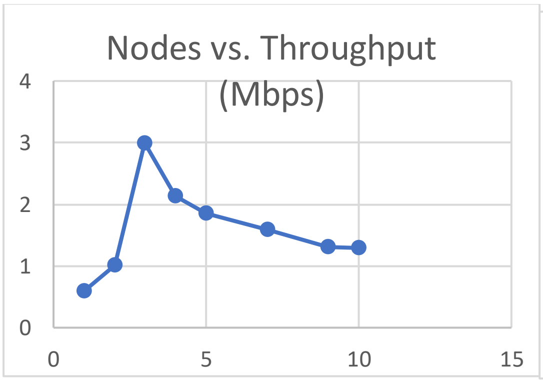
- Application method : Unicast
- Application type : Custom
- Source ID : 1
- Destination ID : 2
- Packet size : Exponential, 1472 bytes
- Inter arrival time : Exponential, 20000us

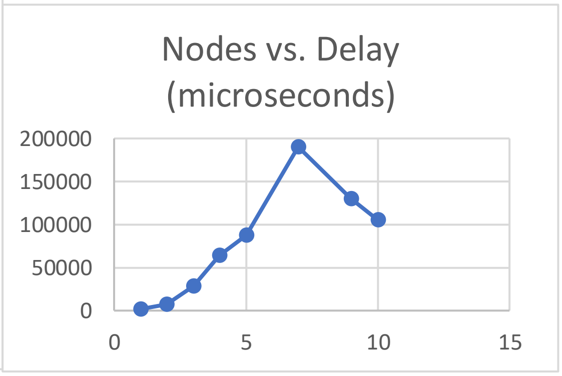
- Click on ‘run simulation’ for 10 seconds
- Obtain the values of throughput and total number of packets transmitted.

1. PURE ALOHA:

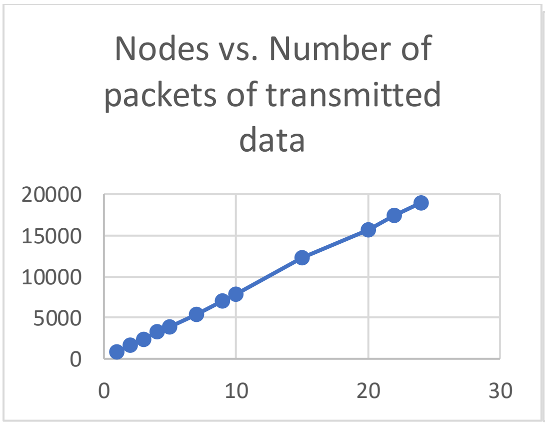


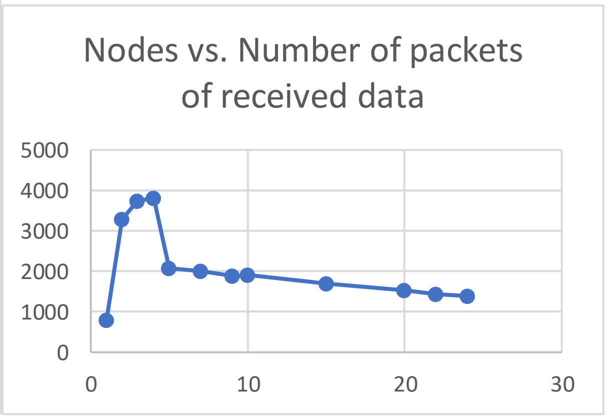


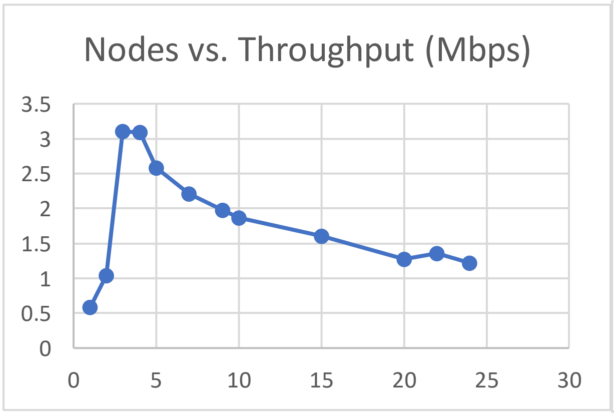


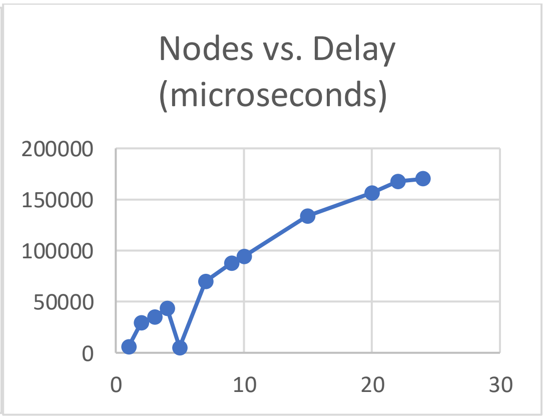


2. Slotted ALOHA







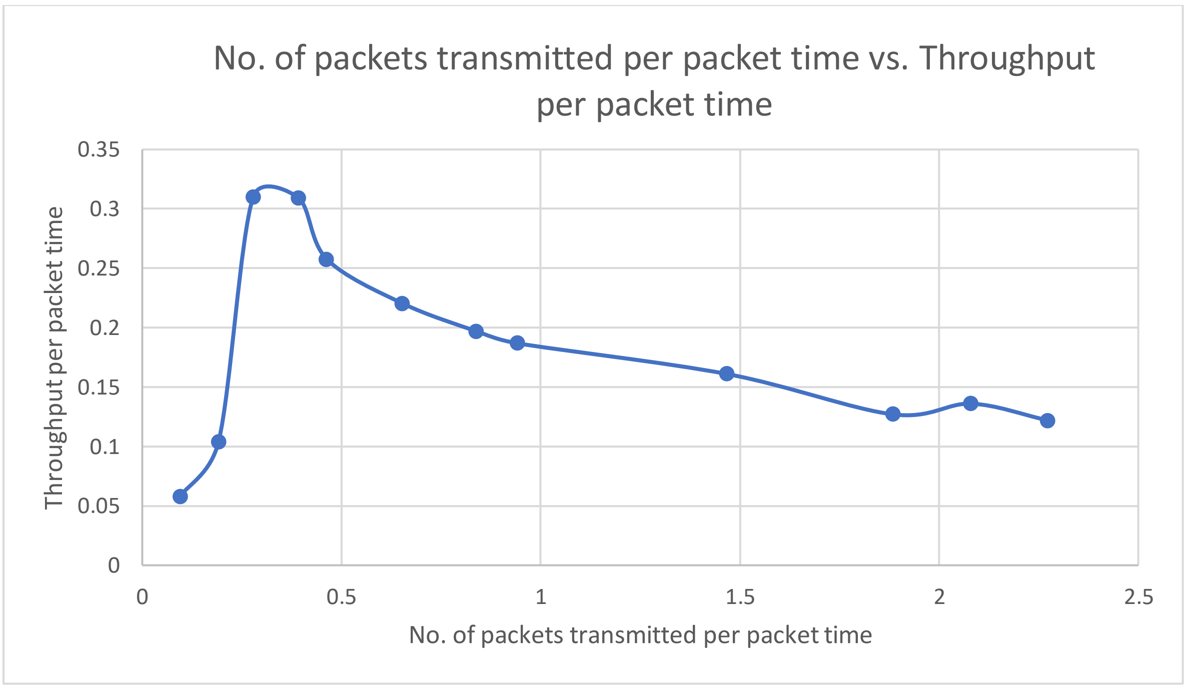


- Select New → Legacy Networks → Slotted ALOHA.
- Click and drop required number of nodes.
- Right click on the grid environment and select the channel characteristics as no path loss.
- Click and drop the appropriate icon and set the following properties as,

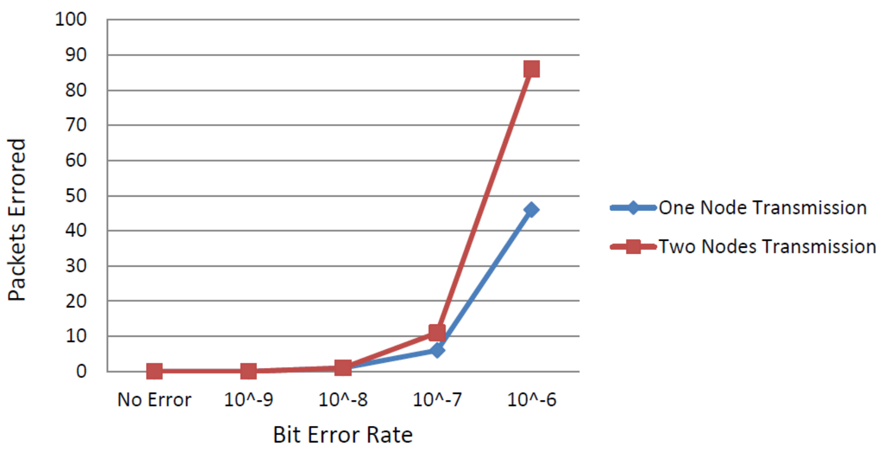
Application method:	Unicast
Application type:	Custom
Source ID:	1
Destination ID:	2
Packet Size:	Exponential, 1472 bytes
Inter Arrival time:	Exponential, 20000 µs

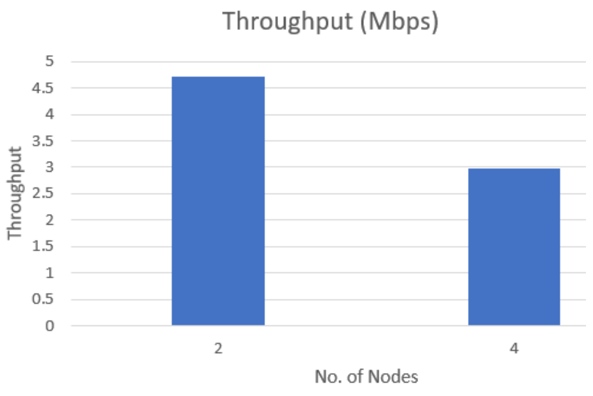
3. CSMA/ CD:

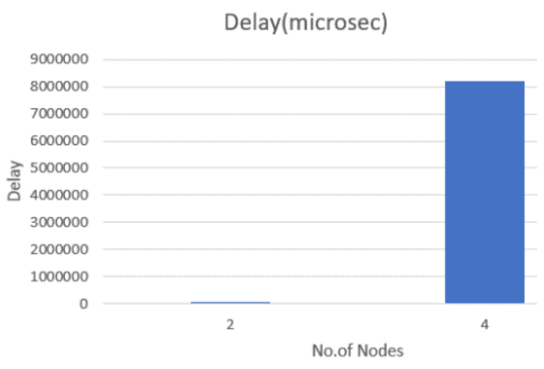
- Select new legacy networks CSMA/CD.
- Place required number of nodes and one hub are to be placed.
- Vary persistence from 1/2,1/3, … . , in all wired nodes to generate other experiments.
- Create broadcast application for all wired nodes and set the following properties.
- Application method: Broadcast
- Application type: Custom
- Packet size: Constant 1472
- Inter arrival time: Exponential 2500
- Run simulator for 10 Seconds.
- Obtain the throughput for various persistent values.


No. of packets transmitted per packet time vs. Throughput per packet time
Throughput per packet time
No. of packets transmitted per packet time
0
0.05
0.1
0.15
0.2
0.25
0.3
0.35
0
0.5
1
1.5
2
2.5

4. CSMA/ CD:


Packets Errored
Bit Error Rate
No Error
10^-9
10^-8
10^-7
10^-6
One Node Transmission
Two Nodes Transmission
0
10
20
30
40
50
60
70
80
90
100


Throughput (Mbps)
Throughput
No. of Nodes
2
4
0
0.5
1
1.5
2
2.5
3
3.5
4
4.5
5


Delay(microsec)
Delay
No.of Nodes
2
4
0
1000000
2000000
3000000
4000000
5000000
6000000
7000000
8000000
9000000

RESULT:

Thus Pure Aloha, Slotted ALOHA, CSMA/CD and protocols of MAC layer are analyzed using NETSIM and throughput for all protocols are calculated and computed.

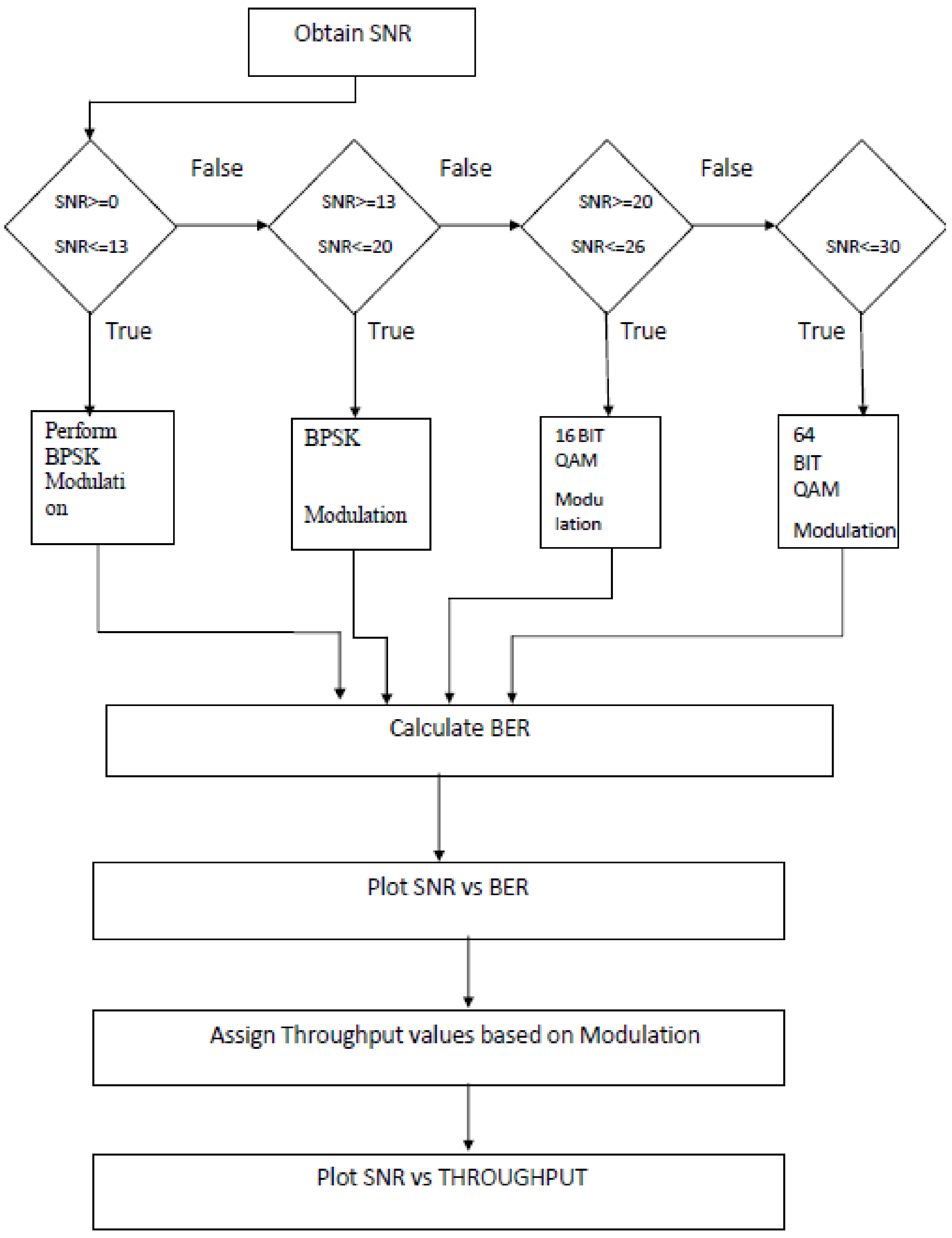
INFERENCE:

It is inferred that though we have to wait to send packets in slotted ALOHA, the throughput of slotted ALOHA is high. In CSMA/CD, as persistence decreases, throughput increases. As the number of nodes increases (2 □4 □ 6) the throughput of the network increases because the channel is able to handle additional network traffic. However, if the number of nodes is increased further, the throughput may decrease as the network traffic is too high, and this leads to collisions. As the number of nodes increases, the delay also increases as it takes more time for a packet to reach its destination.

CONCLUSION:

Thus 3 different MAC protocols have been studied and it can be concluded that slotted aloha has greater throughput than pure aloha but we have to wait to send the packets.For CSMA/CD as no. of nodes increases, the throughput increases.

FLOWCHART:


Obtain SNR
SNR>=0
SNR<=13
False
SNR>=13
SNR<=20
False
SNR>=20
SNR<=26
False
SNR<=30
True
True
True
True
Perform BPSK Modulation
BPSK Modulation
16 BIT QAM Modulation
64 BIT QAM Modulation
Calculate BER
Plot SNR vs BER
Assign Throughput values based on Modulation
Plot SNR vs THROUGHPUT

EXP.NO: 3 DATE: 28.3.2021	ADAPTIVE MODULATION

AIM:

To perform Adaptive modulation and coding and observe its characteristics.

SOFTWARE REQUIRED:

MATLAB R2017a

THEORY:

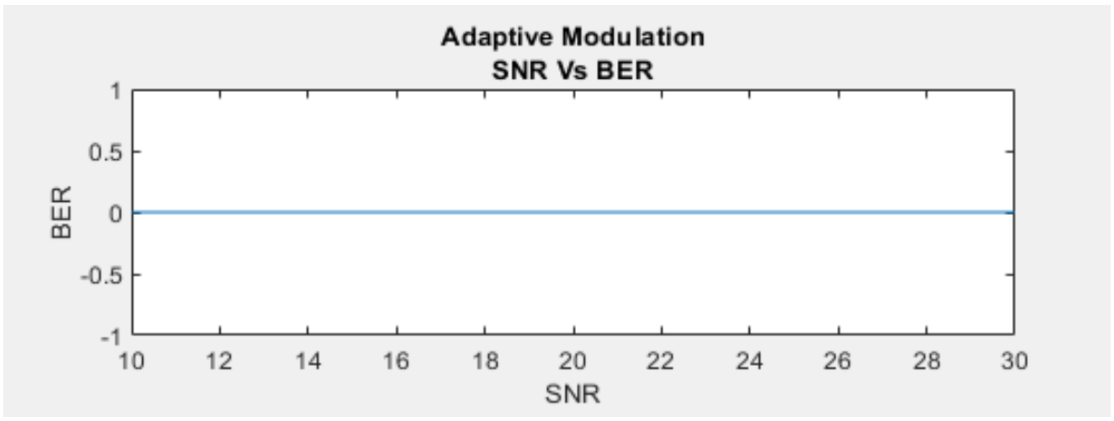
Adaptive Modulation is a technique which allows a radio to change its speed (modulation rate) as conditions in the radio network change. Interference from outside sources, such as changes in the environment (temperature, tree foliage, moving objects) all effect radio coverage. Adaptive modulation enables robust and spectrally efficient transmission over time varying channels. The basic premise is to estimate the channel at the receiver and feed this channel back to the transmitter so that the transmission can be adapted relative to channel characteristics. Thus the system can be designed efficiently for the worst case channel characteristics.

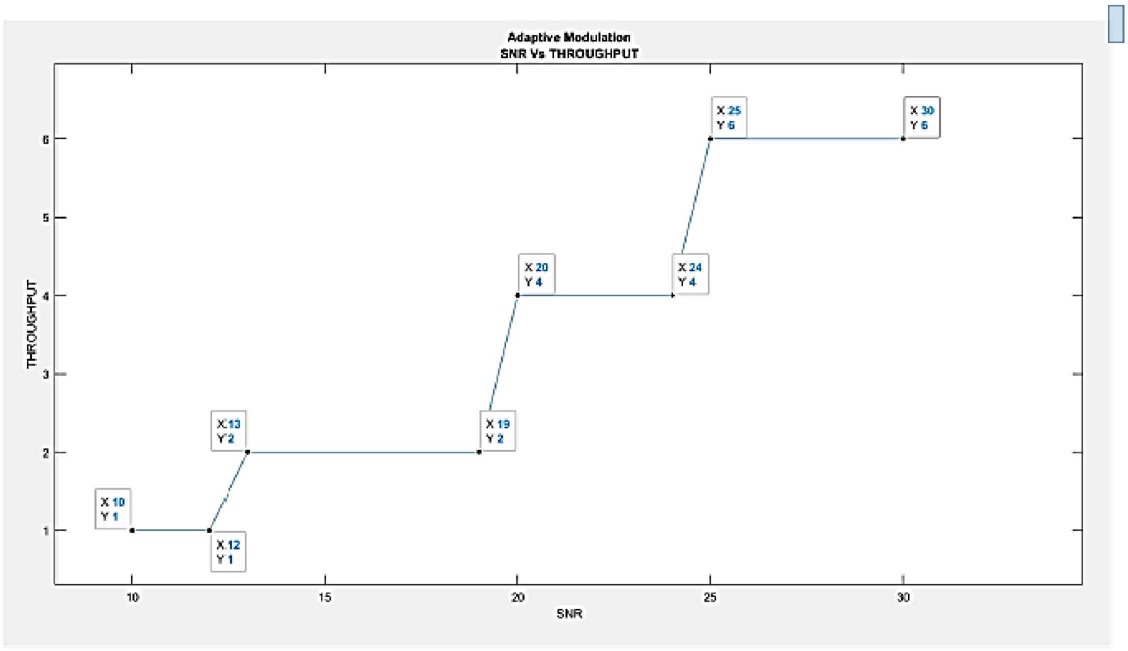
ALGORITHM:

1) Range of SNR for each modulation technique is calculated such that it provides a reliable transmission (minimum BER of 10^-7)

2) Random signal is generated , modulated and demodulated based on SNR and BER is calculated

3) The Throughput is assigned based on the modulation and demodulation technique


Adaptive Modulation
SNR Vs BER
BER
SNR
1
0.5
0
-0.5
-1
10
12
14
16
18
20
22
24
26
28
30


Adaptive Modulation
SNR Vs THROUGHPUT
THROUGHPUT
SNR
X 10
Y 1
X 12
Y 1
X 13
Y 2
X 19
Y 2
X 20
Y 4
X 24
Y 4
X 25
Y 6
X 30
Y 6
1
2
3
4
5
6
10
15
20
25
30

CODE:

```
clc;close all;
snr= randi([10,30],1,15); snr=sort(snr); err=[];
for i=1:length(snr)
if(snr(i)>=0 && snr(i)<13)
a=randi([0,1],1,1000);b=pskmod(a,2);
end
if(snr(i)>=13 && snr(i)<20)
a=randi([0,3],1,1000); b=pskmod(a,4);
end
if(snr(i)>=20 && snr(i)<26)
a=randi([0,15],1,1000); b=qammod(a,16);
end
if(snr(i)>=26 && snr(i)<=30)
a=randi([0,63],1,1000);b=qammod(a,64);
end
N0=1/10^(snr(i)/10);
g=awgn(b,snr(i));
n=sqrt(N0/2)*(randn(1,length(a))+
1i*randn(1,length(a)));
f= g+n;
if(snr(i)>=10 && snr(i)<13)
d=pskdemod(f,2);
 end
if(snr(i)>=13 && snr(i)<20)
d=pskdemod(f,4);
end
if(snr(i)>=20 && snr(i)<26)
d=qamdemod(f,16);
end
if(snr(i)>=26 && snr(i)<=30)
d=qamdemod(f,64);
end
[n,r]=biterr(a,d);
if(r==0)
r=1e-7; end
err=[err r];
end
subplot(2,1,1);semilogy(snr,err,'linewidth',1);
xlabel('SNR');
```

```
ylabel('BER'); title({'AMC';'SNR Vs
BER'});thruput=[];
for i=1:length(snr)
if(snr(i)>=10 && snr(i)<13)
thruput=[thruput 1]; end
if(snr(i)>=13 && snr(i)<20)
thruput=[thruput 2]; end
if(snr(i)>=20 && snr(i)<=25)
thruput=[thruput 4]; end
if(snr(i)>=26 && snr(i)<=30)
thruput=[thruput 6];
end
end
subplot(2,1,2); plot(snr ,thruput);
xlabel('snr'); ylabel('throughput');
title({'AMC';'SNR Vs throughput'});
```

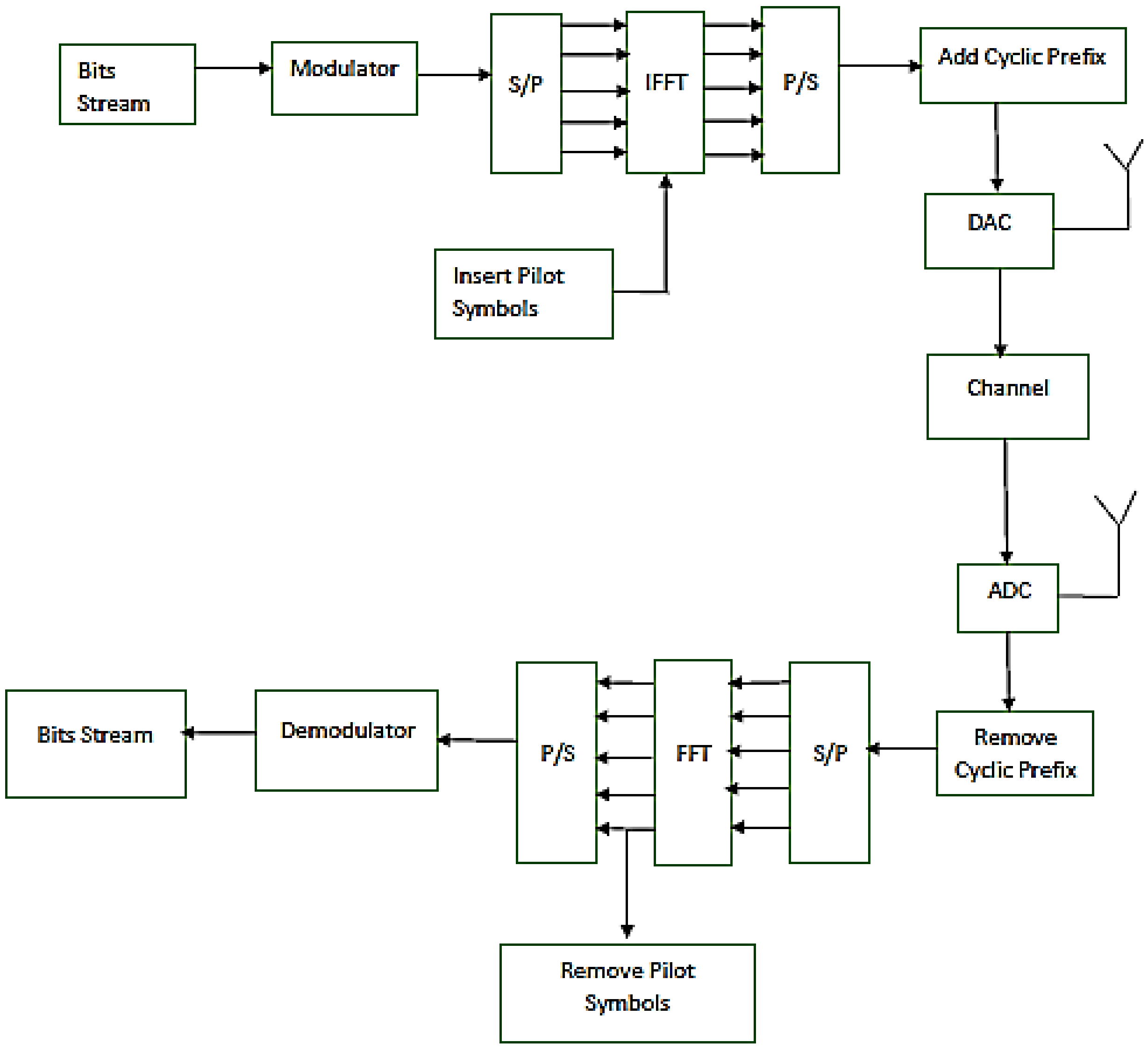
RESULT:

Thus Adaptive Modulation and coding has been implemented and analyzed using MATLAB.

INFERENCE:

- When the SNR is low, BER is high and hence Throughput is low.
- When the SNR is high, BER is low and hence Throughput is high.

Block Diagram:



EXP.NO: 4 DATE: 28.3.2022	**Multicarrier Modulation (OFDM)**

AIM:

To study the characteristics of orthogonal frequency division multiplexing using MATLAB and to plot the BER vs SNR graph.

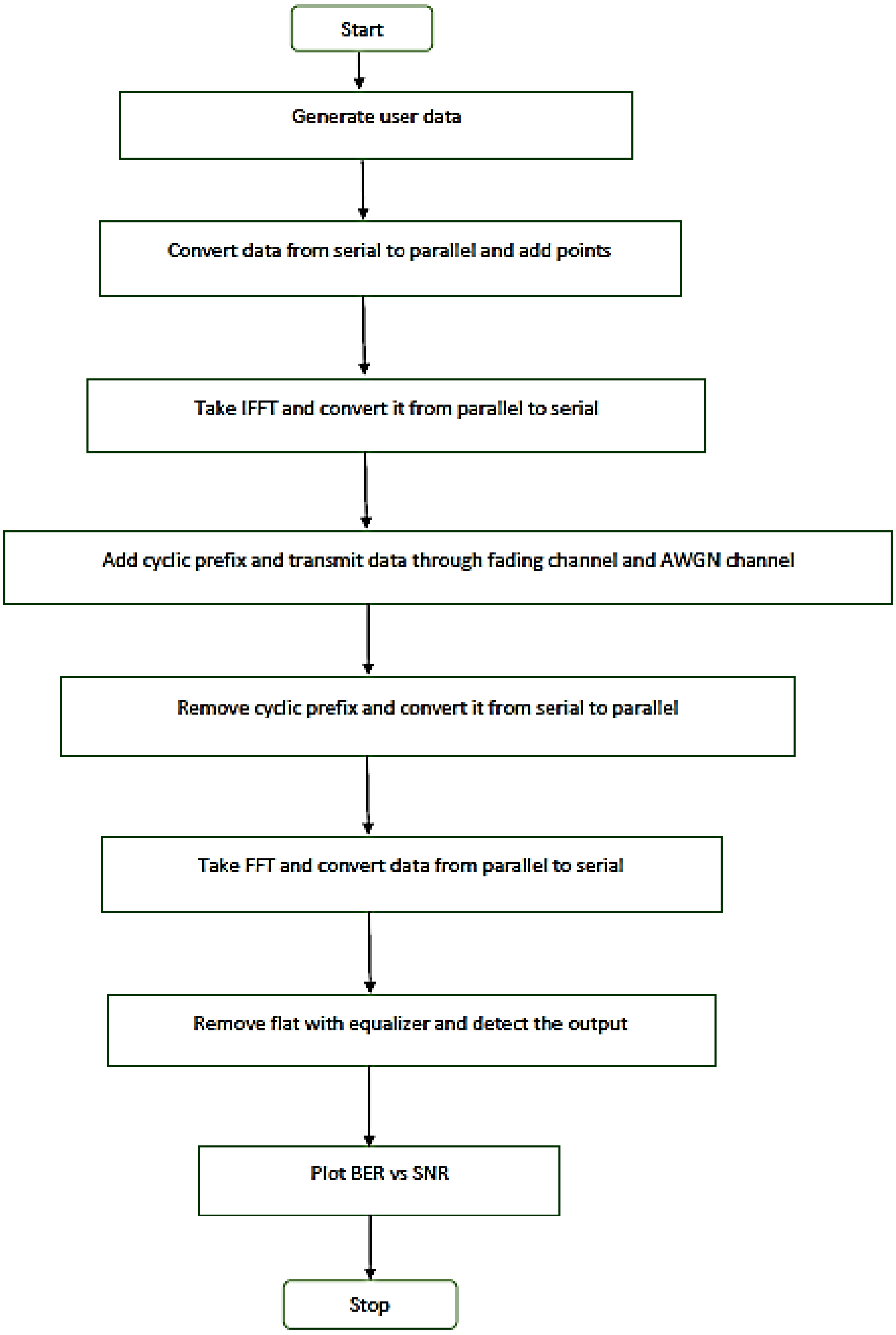
SOFTWARE REQUIRED:

MATLAB R2017a

THEORY:

- Orthogonal FDM (OFDM) spread spectrum technique distributes data over a large number of carriers that are spaced apart at precise frequencies.
- This spacing provides the orthogonality in this technique which prevents the demodulators from seeing frequencies other than their own.
- The benefits of OFDM are high spectral efficiency, resiliency to RF interference, and lower multi-path distortion.
- OFDM is currently the basis of the physical layer of the major wireless systems, such as WiMAX IEEE 802.11n, but it can be also found in cable technology systems such as DSL. There are several variants of OFDM such as VOFDM (vector) or COFDM (coded).

Flow chart:


Start
Generate user data
Convert data from serial to parallel and add points
Take IFFT and convert it from parallel to serial
Add cyclic prefix and transmit data through fading channel and AWGN channel
Remove cyclic prefix and convert it from serial to parallel
Take FFT and convert data from parallel to serial
Remove flat with equalizer and detect the output
Plot BER vs SNR
Stop

ALGORITHM:

- Generate user data.
- Convert data from serial to parallel and add points.
- Take IFFT and convert it from parallel to serial.
- Add cyclic prefix and transmit data through fading channel and AWGN channel.
- Remove cyclic prefix and convert it from serial to parallel.
- Take FFT and convert data from parallel to serial.
- Remove flat with equalizer and detect the output.
- Plot BER vs SNR.

CODE:

```
clc; close all;
x=randi([0 1],1,4096); snr=0:30; biterror=[];
for i =1:4
 y=pskmod(x,2^i);
 if(i==3)
 y=qammod(x,2^(i+1));
 end
 if(i==4)
 y=qammod(x,2^(i+2));
 end
 p=reshape(y,64,64); q=ifft(p,64); s=reshape(q,1,4096);
 be=[];
 for j=0:1:30
h=1/sqrt(rand(1,1)+i*sqrt(rand(1,1))); r=h*s;
n=awgn(r,j,'measured'); m=inv(h)*n; p11=reshape(m,64,64);
 q11=fft(p11,64); s11=reshape(q11,1,4096);
y11=pskdemod(s11,2^i);
 if(i==3)
y11=qamdemod(s11,2^(i+1));
end
```

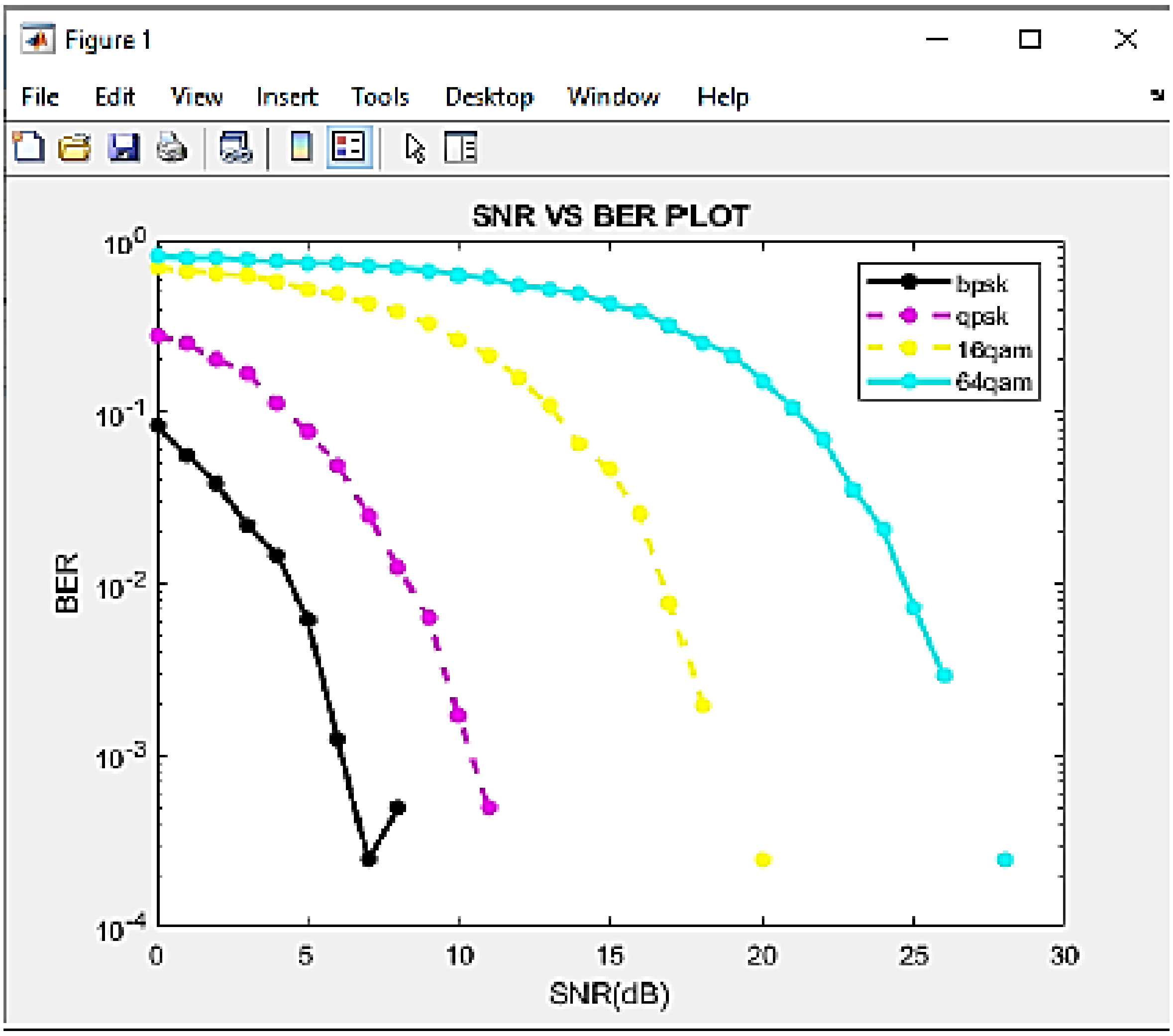


Figure 1
File
Edit
View
Insert
Tools
Desktop
Window
Help
SNR VS BER PLOT
BER
SNR(dB)
bpsk
qpsk
16qam
64qam
10^0
10^{-1}
10^{-2}
10^{-3}
10^{-4}
0
5
10
15
20
25
30

```
if(i==4)
 y11=qamdemod(s11,2^(i+2));
 end
 [num1,e1]=symerr(y11,x);
 be=[be e1];
 end
 biterror(i,:)=be;
end
semilogy(snr, biterror(1,:),'*-k','linewidth',2);hold on;
semilogy(snr, biterror(2,:),'*--m','linewidth',2);hold
on;
semilogy(snr, biterror(3,:),'*--y','linewidth',2);hold
on;
semilogy(snr, biterror(4,:),'*-c','linewidth',2);hold on;
xlabel('SNR(dB)');ylabel('BER');title('SNR VS BER PLOT');
legend('bpsk','qpsk','16qam','64qam');
```

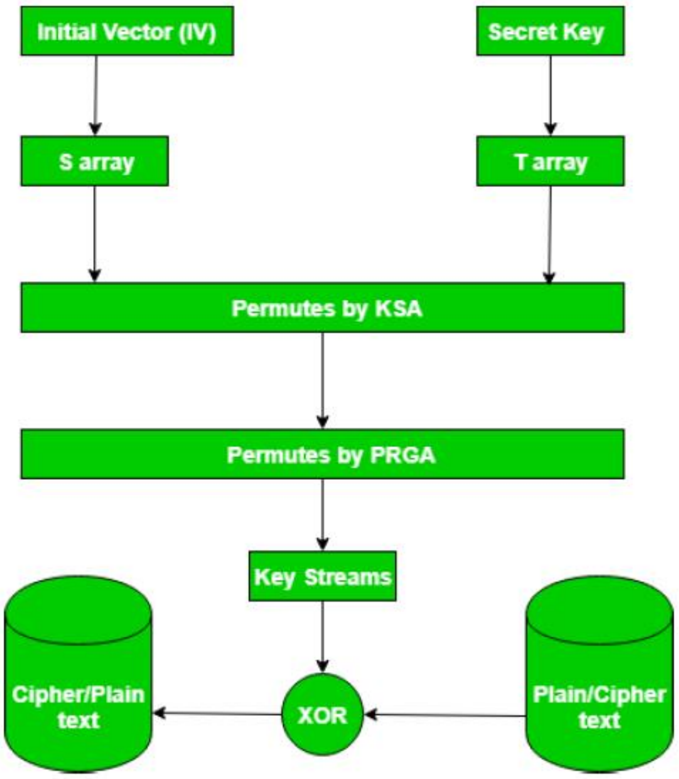
RESULT:

The characteristics of OFDM is studied using MATLAB and BER vs SNR is plotted.

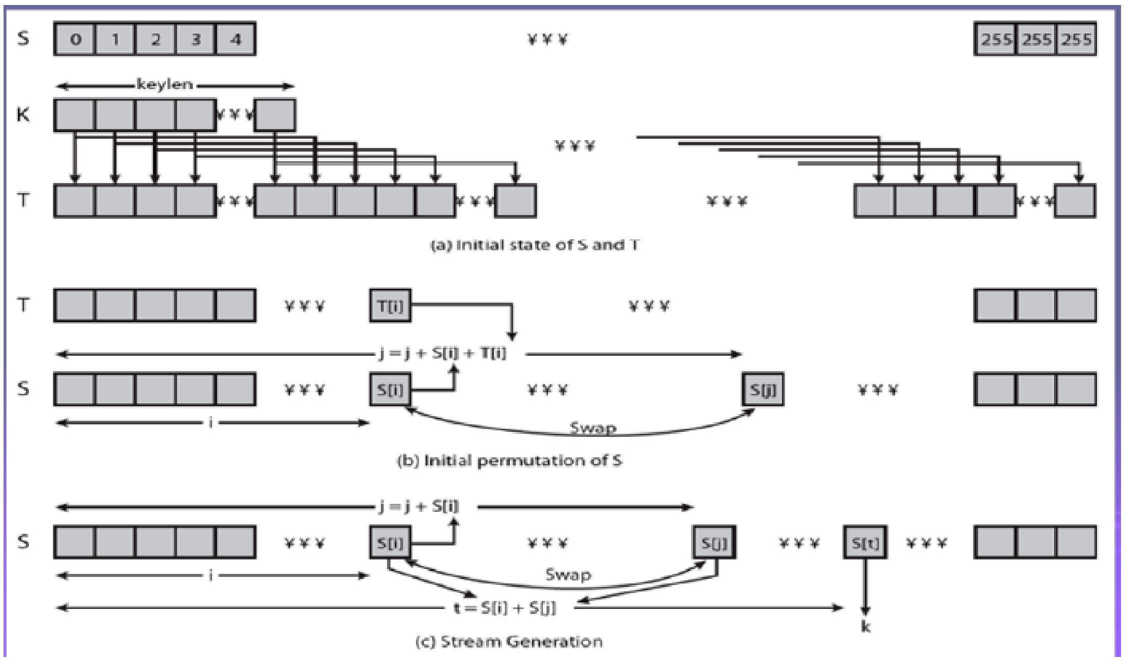
INFERENCE:

The main advantage of OFDM over single-carrier schemes is its ability to cope with severe channel conditions (for example, attenuation at high frequencies in long copper wire, narrowband interference and frequency – selective fading due to multipath) without complex equalization filters.

RC4 FLOW CHART



RC4 DEPICTION



EXP.NO: 5 **DATE: 4.4.2022**	**Network Security Protocol: RC4**

AIM:
To write a MATLAB program for RC4 encryption algorithm, check whether the cipher text can be decrypted to get the plain text again.

SOFTWARE REQUIRED:
MATLAB

THEORY:

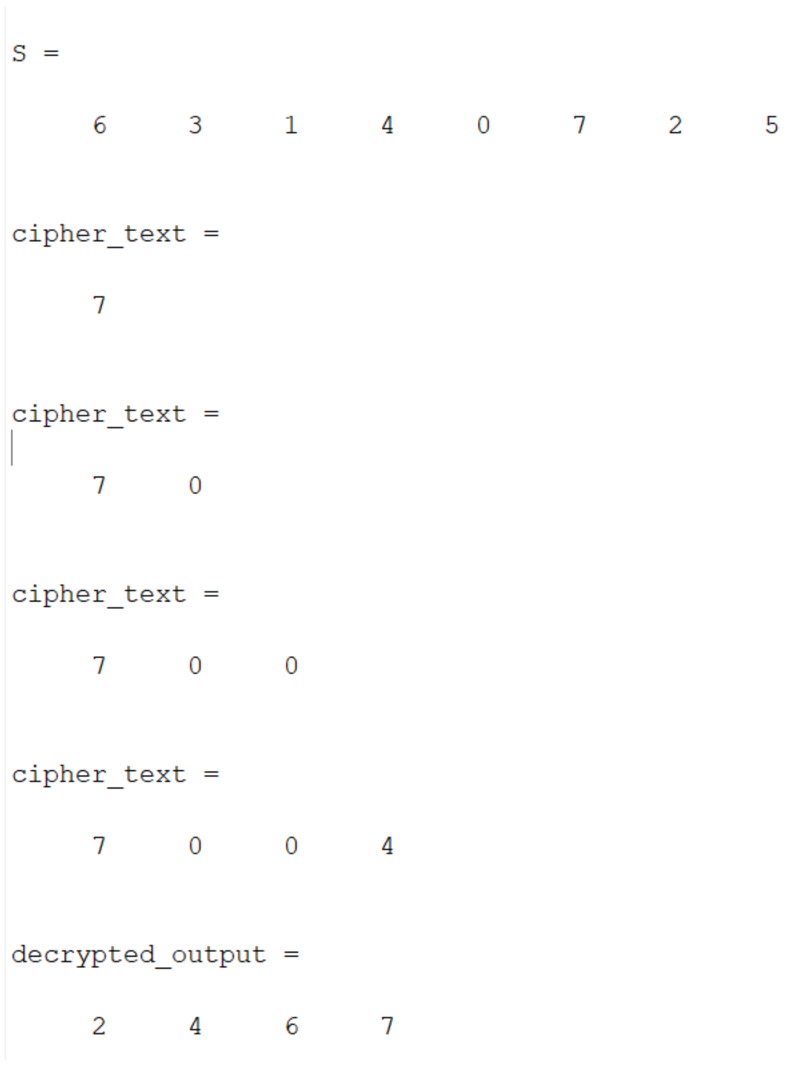
RC4 is a stream cipher and variable length key algorithm. This algorithm encrypts one byte at a time. A key input is a pseudo-random bit generator that produces a stream of 8-bit number that is unpredictable without the knowledge of input key. The output of the generator is called key stream. It is combined one byte at a time with plain text stream cipher using Ex-OR operation.
Key Generation Algorithm:
A variable-length key from 1 to 256 byte is used to initialize a 256-byte state vector S, with elements S[0] to S[255]. For encryption and decryption, a byte k is generated from S by selecting one of the 255 entries in a systematic fashion, then the entries in S are permuted again.

Key-Scheduling Algorithm:
Initialization: The entries of S are set equal to the values from 0 to 255 in ascending order, a temporary vector T, is created.
If the length of the key k is 256 bytes, then k is assigned to T. Otherwise, for a key with length(k-len) bytes, the first k-len elements of T as copied from K and then K is repeated as many times as necessary to fill T. We use T to produce the initial permutation of S. Starting with S[0] to S[255], and for each S[i] algorithm swap it with another byte in S according to a scheme dictated by T[i], but S will still contain values from 0 to 255

Pseudo random generation algorithm (Stream Generation):
Once the vector S is initialized, the input key will not be used. In this step, for each S[i] algorithm swap it with another byte in S according to a scheme dictated by the current configuration of S. After reaching S[255] the process continues, starting from S[0] again

OUTPUT:

Taking K= [1 2 3 4] and message = [2 4 6 7]
Key length = 4



```
S =

     6     3     1     4     0     7     2     5


cipher_text =

     7


cipher_text =

     7     0


cipher_text =

     7     0     0


cipher_text =

     7     0     0     4


decrypted_output =

     2     4     6     7
```

ALGORITHM:

```
/* Initialization */
for i = 0 to 255
do
S[i] = i;
T[i] = K[i mod keylen]
/* Initial Permutation of S */
j = 0;
for i = 0 to 255
do
j = (j + S[i] + T[i]) mod 256;
Swap (S[i], S[j]);
/* Stream Generation */
i, j = 0;
while (true)
i = (i + 1) mod 256;
j = (j + S[i]) mod 256;
Swap (S[i], S[j]);
t = (S[i] + S[j]) mod 256;
k = S[t];
```

CODE:

```
clc;
clear all;
close all;
% RC4 ENCRYPTION ALGORITHM
% INITIALISE A STATE VECTOR
S = zeros(1,8);
% GENERATE A KEY VECTOR
%KEY_STRING = 'ECE'
KEY = [1 2 3 4];
K = zeros(1,8);
% GENERATE A PLAIN TEXT OF MESSAGE
%MSG_STRING = 'WELCOME TO WCN LAB'
MSG = [2 4 6 7];
KEY_LENGTH = length(KEY);
% PERFORM KEY SCHEDULING ALGORITHM
for p = 1:8
 S(p) = p-1;
 TEMP = mod(p-1,KEY_LENGTH)+1;
 K(p) = KEY(TEMP);
%  K(p)
```

```
end
% GENERATE CIPHER TEXT
q = 0;
for p = 1:8
 q = mod((q+S(p)+K(p)),8)+1;
%  q
 S([p q]) = S([q p]);
%  S([p q])
 q=q-1;
end
S
cipher_text = [];
cipher_text_store = [];
key_stream = [];
% PSEUDO RANDOM GENERATION ALGORITHM (PRGA)
q = 0;
for p = 1:length(MSG)
 q = mod((q+S(p+1)),8)+1;
 S([p+1 q]) = S([q p+1]);
%  S([p+1 q])

 TEMP = mod((S(p+1)+S(q)),8);
%  TEMP
 current_key = S(TEMP+1);
%  current_key
 key_stream = [key_stream,current_key];
 cipher_text(p) = bitxor(MSG(p),current_key)
 cipher_text_store = [cipher_text_store,cipher_text];
  q=q-1;
end
%cipher_text_string = cipher_text_store
% RC4 DECRYPTION
decrypted_message = [];
decrypted_msg = [];
for p = 1:length(cipher_text)
 current_message = bitxor(cipher_text(p),key_stream(p));
 decrypted_message = [decrypted_message,current_message];
end
decrypted_output = decrypted_message
```

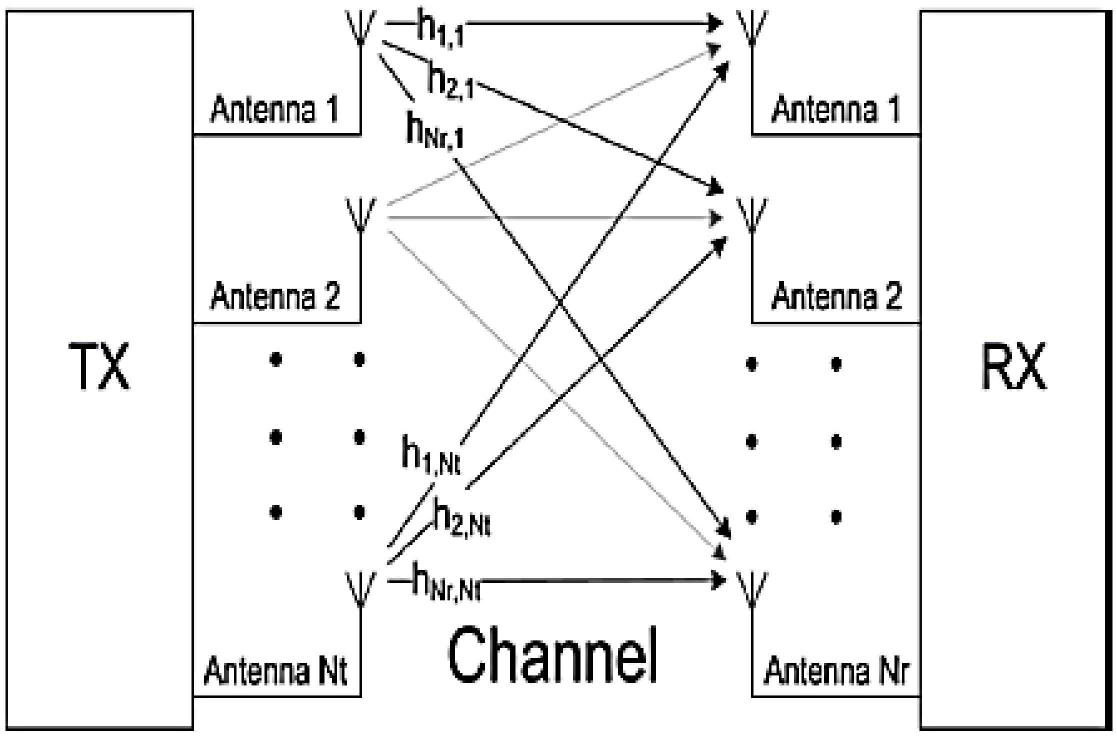
RESULT:

RC4 algorithm has been simulated and implemented using MATLAB and the encryption and decryption has been verified.

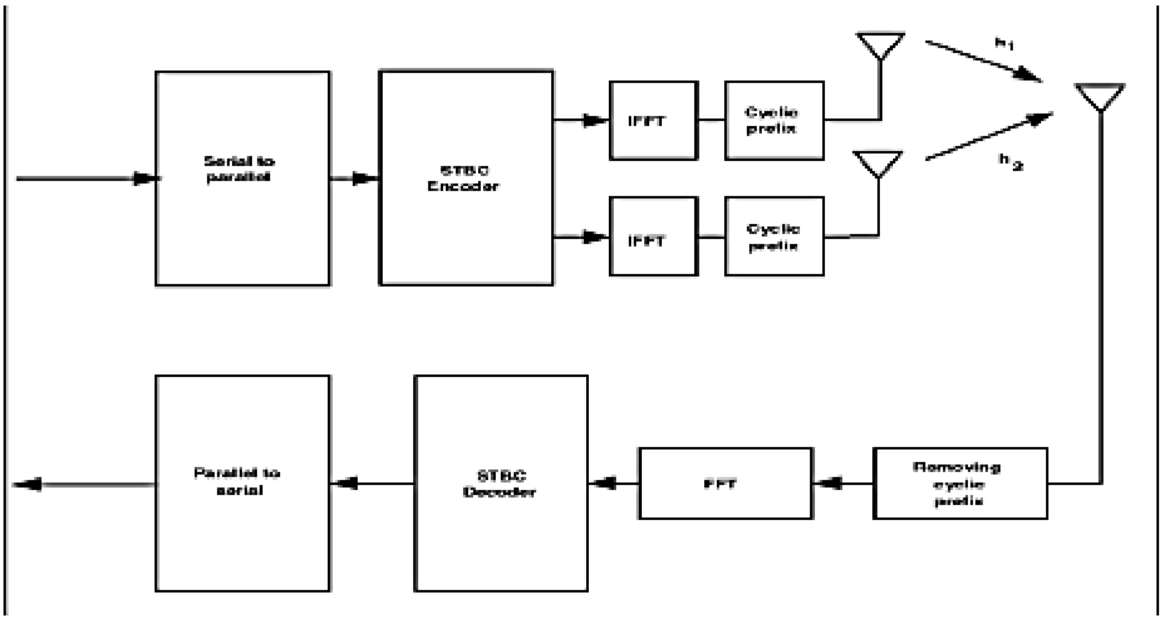
INFERENCE:

RC4 is a stream cipher that is simple to use and the speed of operation is also higher. It doesn't require lots of memory and can also be implemented on large streams of data. But, one in every 256 keys is a weak key and thus RC4 is vulnerable and insecure at times due to which very few applications use it nowadays.

BLOCK DIAGRAM:
MIMO:



STBC:



EXP.NO: 6 DATE: 23.4.2022	Space Time Block Codes

AIM:
To study Space Time Block Codes, using MATLAB.

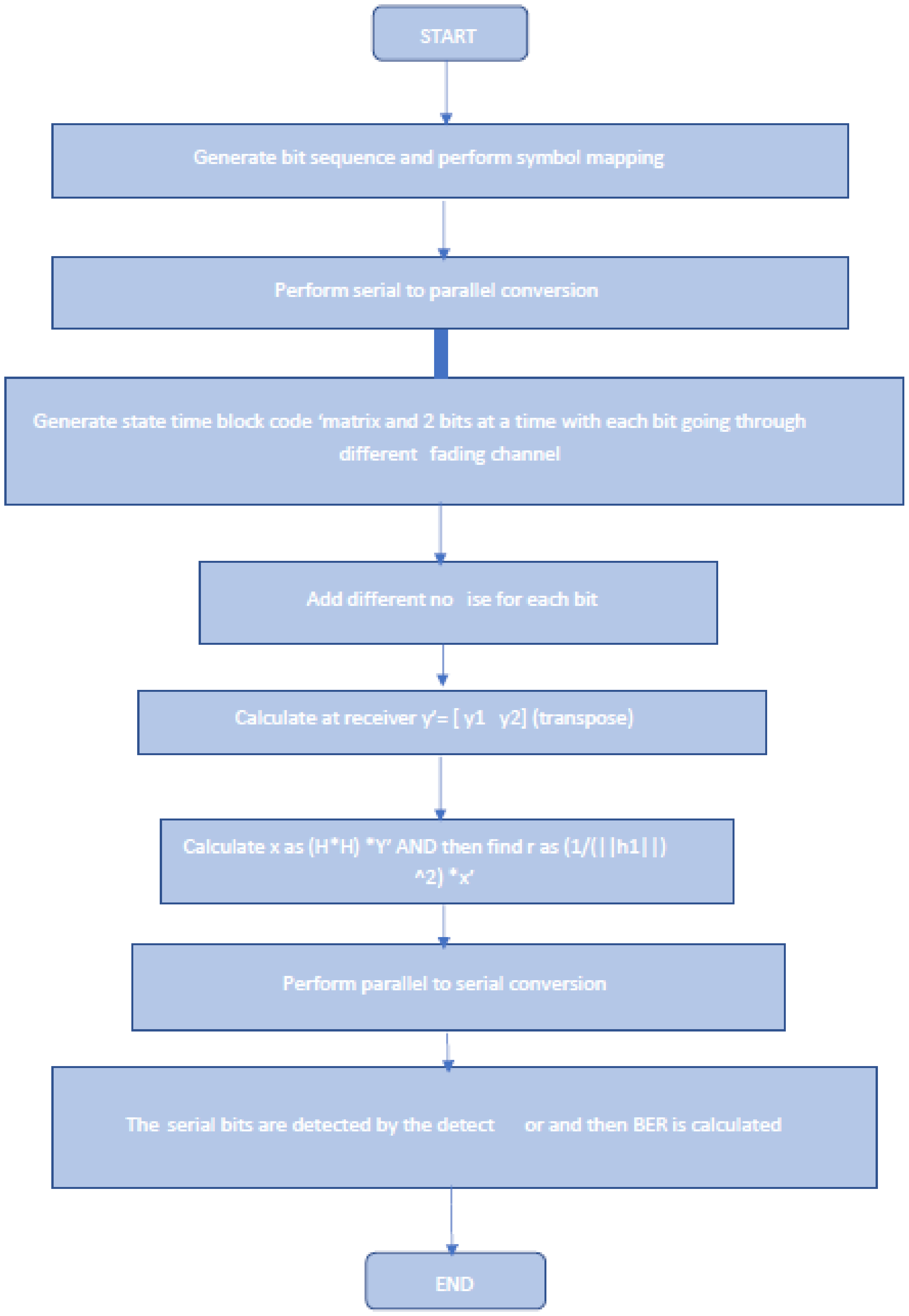
SOFTWARE USED:
MATLAB software.

THEORY:

MIMO TECHNOLOGY:
MIMO stands for Multiple-Input Multiple-Output, is a wireless technology that uses multiple transmitters and receivers to transfer more data at the same time. All wireless products with 802.11n support MIMO. The technology helps allow 802.11n to reach higher speeds than products without 802.11n. MIMO technology uses a natural radio-wave phenomenon called multipath. With multipath, transmitted information bounces off walls, ceilings and other objects, reaching the receiving antenna multiple times at different angles and slightly different times. With multipath, MIMO technology uses multiple, smart transmitters and receivers with an added spatial dimension, increasing performance and range. It increases receiver signal-capturing power by enabling antennas to combine data streams arriving from different paths at different times.
It introduces signalling degrees of freedom that were absent in SISO system. This is referred to as the spatial degree of freedom. The spatial degrees of freedom can either be exploited for "Diversity" or "Multiplexing" or a combination of the two. When antennas out number spatial streams, the antennas can add receiver diversity and increase range. In simple terms, diversity means redundancy. Diversity can also be achieved using multiple transmit antennas by using Space Time Coding (STC) techniques.

SPACE-TIME BLOCK CODING:
Space-time block codes (STBC) are a generalized version of Alamouti scheme. These schemes have the same key features. Therefore, these codes are orthogonal and can achieve full transmit diversity specified by the number of transmit antennas. In another word, spacetime block codes are a complex version of Alamouti's space-time code in, where the encoding and decoding schemes are the same as there in the Alamouti space-time code in both the transmitter and receiver sides.

FLOW CHART:


START
Generate bit sequence and perform symbol mapping
Perform serial to parallel conversion
Generate state time block code 'matrix and 2 bits at a time with each bit going through different fading channel
Add different no ise for each bit
Calculate at receiver y'= [y1 y2] (transpose)
Calculate x as (H*H) *Y' AND then find r as (1/(| |h1| |) ^2) *x'
Perform parallel to serial conversion
The serial bits are detected by the detect or and then BER is calculated
END

The data are constructed as a matrix which has its rows equal to the number of the transmit antennas and its columns equal to the number of the time slots required to transmit the data. At the receiver side, when signals are received, they are first combined and then sent to the maximum likelihood detector where the decision rules are applied.
Although these methods avoid the need for channel estimation, they often suffer from problems such as error propagation. Training-based methods seem to give very good results on the performance of channel estimation at the receiver. Pure training-based schemes can be considered as an advantage when an accurate and reliable MIMO channel needs to be obtained. However, this could also be a disadvantage when bandwidth efficiency is required. This is because pure training-based schemes reduce the bandwidth efficiency considerably due to the use of a long training sequence which is necessarily needed in order to obtain a reliable MIMO channel estimate. Because of the computation complexity of blind and semiblind methods, many wireless communication systems still use pilot sequences to estimate the channel parameters at the receiver side.

ENCODING:
It was designed for a two-transmit antenna system and has the coding matrix:

$$C_2 = \begin{bmatrix} c_1 & c_2 \\ -c_2^* & c_1^* \end{bmatrix}$$

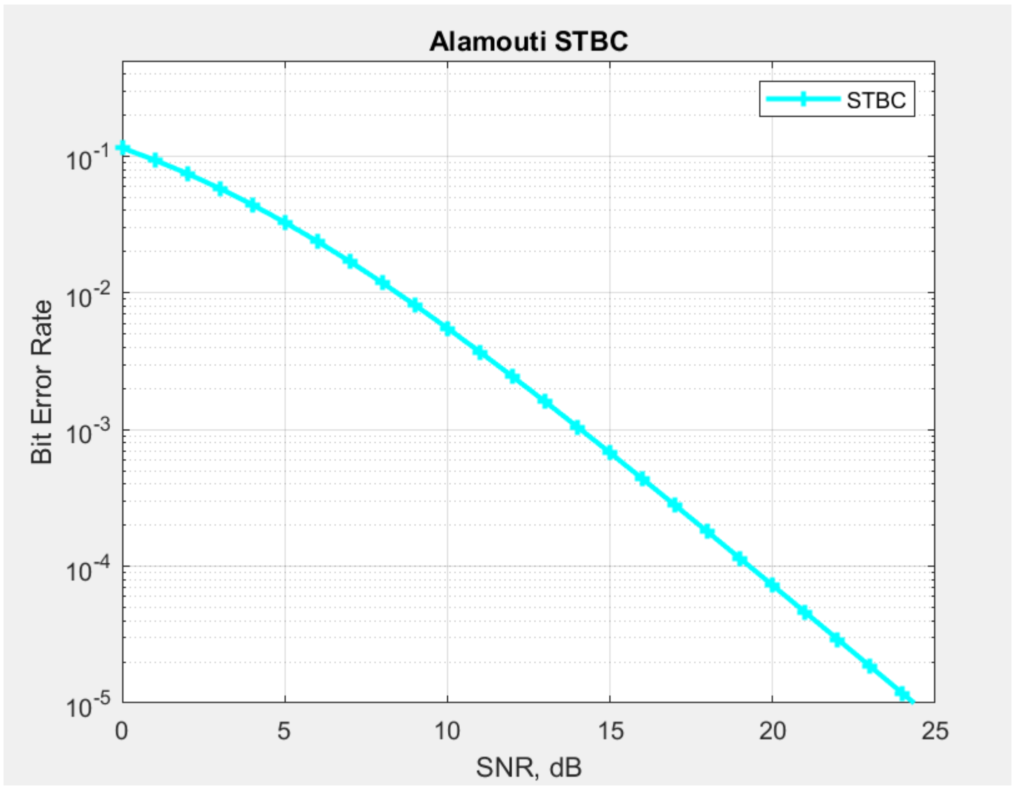
Where * denotes complex conjugate.

Where * denotes complex conjugate.
It is readily apparent that this is a rate-1 code. It takes two time slots to transmit two symbols. Using the optimal decoding scheme discussed below, the Bit Error Rate (BER) of this STBC is equivalent to 2 n R – branch maximal ratio combining (MRC). This is a result of the perfect orthogonality between the symbols after receive processing – there are two copies of each symbol transmitted and n R copies received.

DECODING:

One particularly attractive feature of orthogonal STBCs is that maximum likelihood decoding can be achieved at the receiver with only linear processing. In order to consider a decoding method, a model of the wireless communications system is needed.

OUTPUT:


Alamouti STBC
STBC
Bit Error Rate
SNR, dB
10^-1
10^-2
10^-3
10^-4
10^-5
0
5
10
15
20
25

ALGORITHM:
1. Generate random binary sequence of +1s and -1s.
2. Group them into pairs of two symbols.
3. Code it as per the Alamouti Space Time Code, multiply the symbols with the channels and then add AWGN.
4. Equalize the received symbols.
5. Perform hard decision decoding and count the bit errors.
6. Repeat for multiple values and plot the simulation.

Code:

```
clc
clear all
close all
N = 10^6; % number of bits or symbols
SNR = [0:25];
for ii = 1:length(SNR)
ip = rand(1,N)>0.5;
s = 2*ip-1;
sCode = zeros(2,N);
sCode(:,1:2:end) = (1/sqrt(2))*reshape(s,2,N/2);
sCode(:,2:2:end) = (1/sqrt(2))*(kron(ones(1,N/2),[-
1;1]).*flipud(reshape(conj(s),2,N/2)));
h = 1/sqrt(2)*(randn(1,N) + 1i*randn(1,N));
hMod = kron(reshape(h,2,N/2),ones(1,2));
n = 1/sqrt(2)*(randn(1,N) + 1i*randn(1,N));
y = sum(hMod.*sCode,1) + 10^(-SNR(ii)/20)*n;
yMod = kron(reshape(y,2,N/2),ones(1,2));
yMod(2,:) = conj(yMod(2,:));
hEq = zeros(2,N);
hEq(:,[1:2:end]) = reshape(h,2,N/2);
hEq(:,[2:2:end]) = kron(ones(1,N/2),[1;-
1]).*flipud(reshape(h,2,N/2));
hEq(1,:) = conj(hEq(1,:));
hEqPower = sum(hEq.*conj(hEq),1);
yHat = sum(hEq.*yMod,1)./hEqPower;
yHat(2:2:end) = conj(yHat(2:2:end));
% receiver - hard decision decoding
ipHat = real(yHat)>0;
% counting the errors
nErr(ii) = size(find([ip- ipHat]),2);
```

```
end
simBer = nErr/N; % simulated ber
SNRLin = 10.^(SNR/10);
pAlamouti = 1/2 - 1/2*(1+2./SNRLin).^(-1/2);
berAlamouti = pAlamouti.^2.*(1+2*(1-pAlamouti));
close all
figure
semilogy(SNR,berAlamouti,'c+-','LineWidth',2);
hold on
axis([0 25 10^-5 0.5])
grid on
legend('STBC');
xlabel('SNR, dB');
ylabel('Bit Error Rate');
title('Alamouti STBC');
```

RESULT:

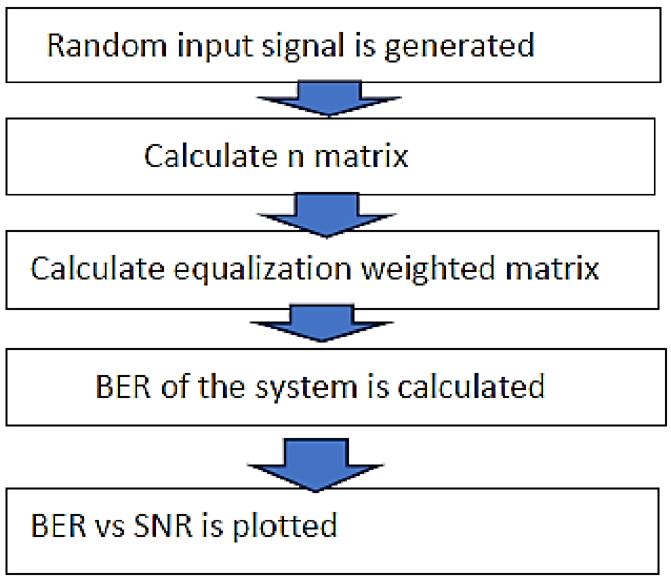
Thus, the Space Time Block Code using Alamouti coding scheme is studied and the Bit Error Rate is analysed.

INFERENCE:

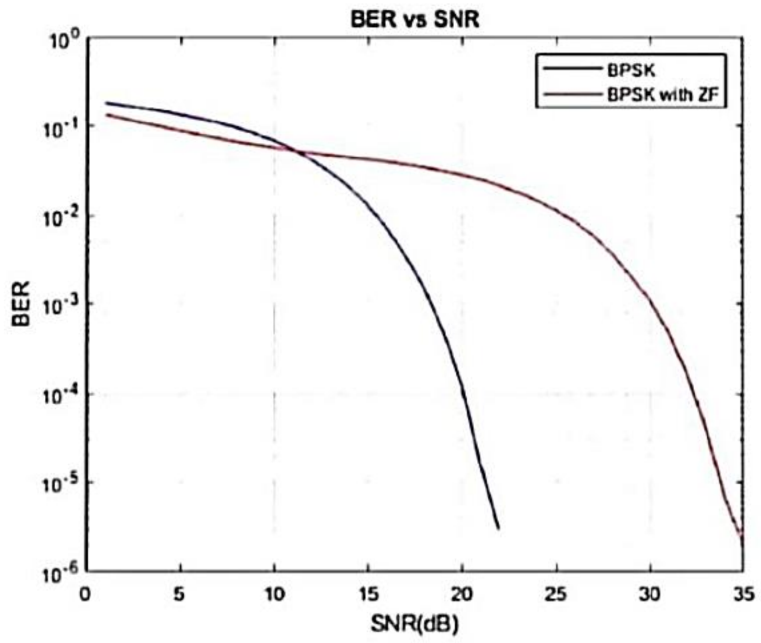
1. Channel response of transmitter antenna is independent of each other.
2. The fact that HH. Y is a diagonal matrix ensured that there is no cross talk between x1, x2.
3. With Alamouti STBC, we are transmitting from two antennas. Hence the total power transmitted is higher than any other schemes.
4. The signal to the noise ratio is getting reduced while using STBC coding.

FLOW CHART:

ZERO FORCING:


Random input signal is generated
Calculate n matrix
Calculate equalization weighted matrix
BER of the system is calculated
BER vs SNR is plotted

OUTPUT FOR ZERO FORCING:



EXP.NO: 7 DATE: 23.4.2022	**Equalization techniques for wireless channels**

AIM:
To implement equalizers using ZF &LMS algorithm.

SOFTWARE USED:
MATLAB R2020a

THEORY:

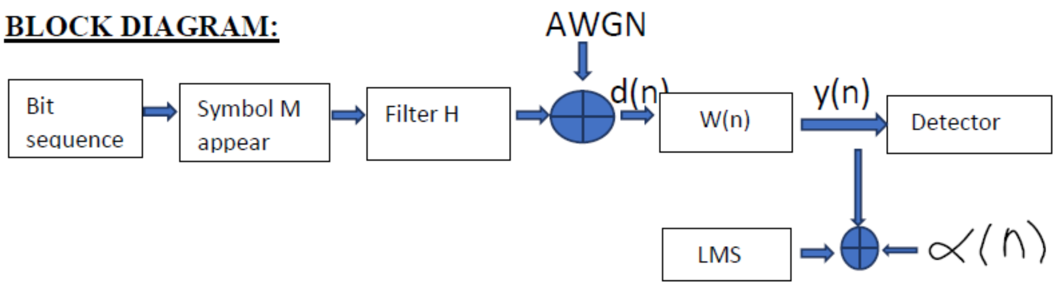
1. **ZERO FORCING:**
It's a linear equalizer which uses universe impulse response to compensate channel effect. An overall impulse response is equal to one for defected symbol and zero for all other received symbols. The zero-forcing equalizer nullified the circumference but does not consider effect of noise. The noise may be increased in this process.

PROCEDURE:
1. Random input signal is generated.
2. Calculate H matrix.
3. Calculate equalization weighted matrix.
4. BER of the system is calculated.
5. BER vs SNR is plotted.

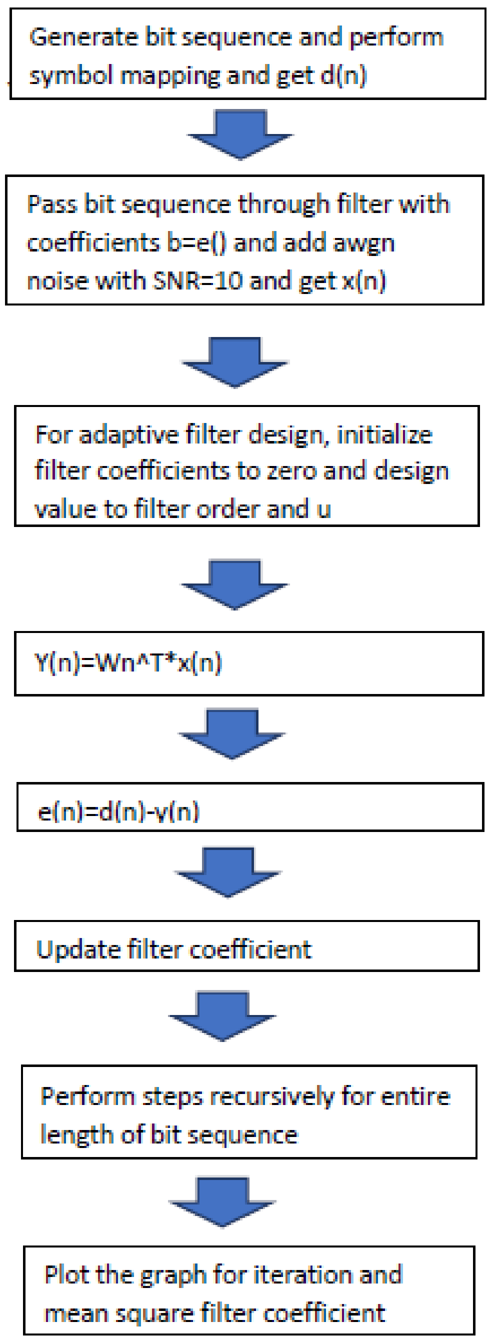
CODE:

```
clc; clear all; close all;
message = randi([0,1],1,100000);
snr = 0:2:40; mod = 2;
L = 2; % No. of TAPS
r = 3; % 3 TAP EQUALISER
modulated_bpsk_msg = pskmod(message,mod);
h1 = 1; h2 = 0.7;
% ORDER OF H MATRIX
H = zeros(r,r+L-1);
% FORMING H MATRIX FOR 3 TAP EQUALISER
for p = 1:r
```

BLOCK DIAGRAM:



LEAST MEAN SQUARE:


Generate bit sequence and perform symbol mapping and get d(n)

Pass bit sequence through filter with coefficients b=e() and add awgn noise with SNR=10 and get x(n)

For adaptive filter design, initialize filter coefficients to zero and design value to filter order and u

Y(n)=Wn^T*x(n)

e(n)=d(n)-y(n)

Update filter coefficient

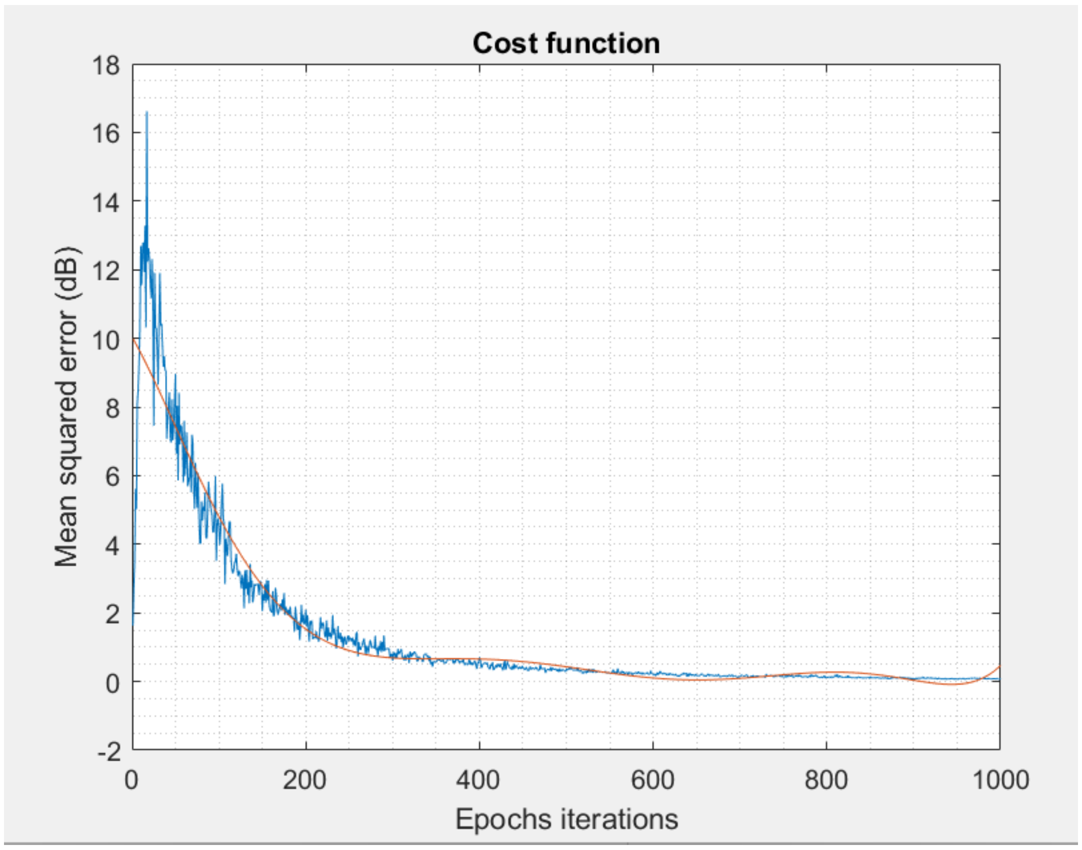
Perform steps recursively for entire length of bit sequence

Plot the graph for iteration and mean square filter coefficient

```
H(p,p:p+L-1) = [h1 h2];
end
% FORMING INPUT MATRIX X
x = []; X = modulated_bpsk_msg;
X_1 = circshift(X,1); X_1(1) = 0;
X1 = circshift(X,-1); X1(end) = 0;
X2 = circshift(X,-2); X2(end-1:end)= 0;
x = [X2;X1;X;X_1];C = ((H*H')\H)*[0;0;1;0];
ber_without_ZFE = [];ber_with_ZFE = [];
for p = 1:length(snr)
y = awgn([h1 h2]*[X;X_1],snr(p),'measured');
Noise = y - ([h1 h2]*[X;X_1]);
Noise_1 = circshift(Noise,-1);
Noise_1(end) = 0;Noise_2 = circshift(Noise,-2);
Noise_1(end-1:end) = 0;

Y = (H*x)+ [Noise_2;Noise_1;Noise];
X_PRIME = C.'*Y;
Demodulated_BPSK_msg_with_ZFE =
pskdemod(X_PRIME',mod);
Demodulated_BPSK_msg_without_ZFE = pskdemod(y',mod);
[number1,ratio1] =
biterr(message,Demodulated_BPSK_msg_with_ZFE');
[number2,ratio2] =
biterr(message,Demodulated_BPSK_msg_without_ZFE');
ber_without_ZFE = [ber_without_ZFE,ratio1];
ber_with_ZFE = [ber_with_ZFE,ratio2];
end
semilogy(snr,ber_without_ZFE)
hold on
semilogy(snr,ber_with_ZFE,'--')
legend('BER WITH ZFE','BER WITHOUT ZFE')
title('SNR VS BER FOR BPSK')
xlabel('SNR');ylabel('BER');
```

OUTPUT FOR LEAST MEAN SQUARE:



2. LEAST MEAN SQUARE:

LMS algorithm is a type of filter used in machine learning. It uses a technique called “ Method of steepest descent” and continuously estimates result by updating filter weights. Through the principles of algorithm convergence, The LMS algorithm provides particular learning curves useful in machine learning theory and implementation. Many of these ideas are part of dedicated work on refining machine learning models, matching i/p s to o/p s, making training and test processors more effective , and generally pursuing “convergence” where the iterative learning process resolves into a final result instead of getting off track.

CODE:

```
clc; clear all; close all;
h = [0.9 0.3 0.5 -0.1]; SNRr = 30;runs = 100;
eta = 5e-3; order=12;
for run = 1 : runs
U = zeros(1,order); W = randn(1,order);N = 1000;
Bits = 2; data = randi([0 1],1,N);
d = real(pskmod(data,Bits)); r = filter(h,1,d);
x = awgn(r, SNRr);
for n = 1 : N
U(1,2:end) = U(1,1:end-1); U(1,1) = x(n); y = (W)*U';
e = d(n) - y; W = W + eta * e * U ; J(run,n) = e *
e';
end
end
MJ = mean(J,1);
figure
plot((MJ))
trendMJ = polyval(polyfit((0:N),[0 (MJ)],7),(1:N));
hold on
plot(trendMJ)
grid minor
xlabel('Epochs iterations');
ylabel('Mean squared error (dB)');
title('Cost function');
```

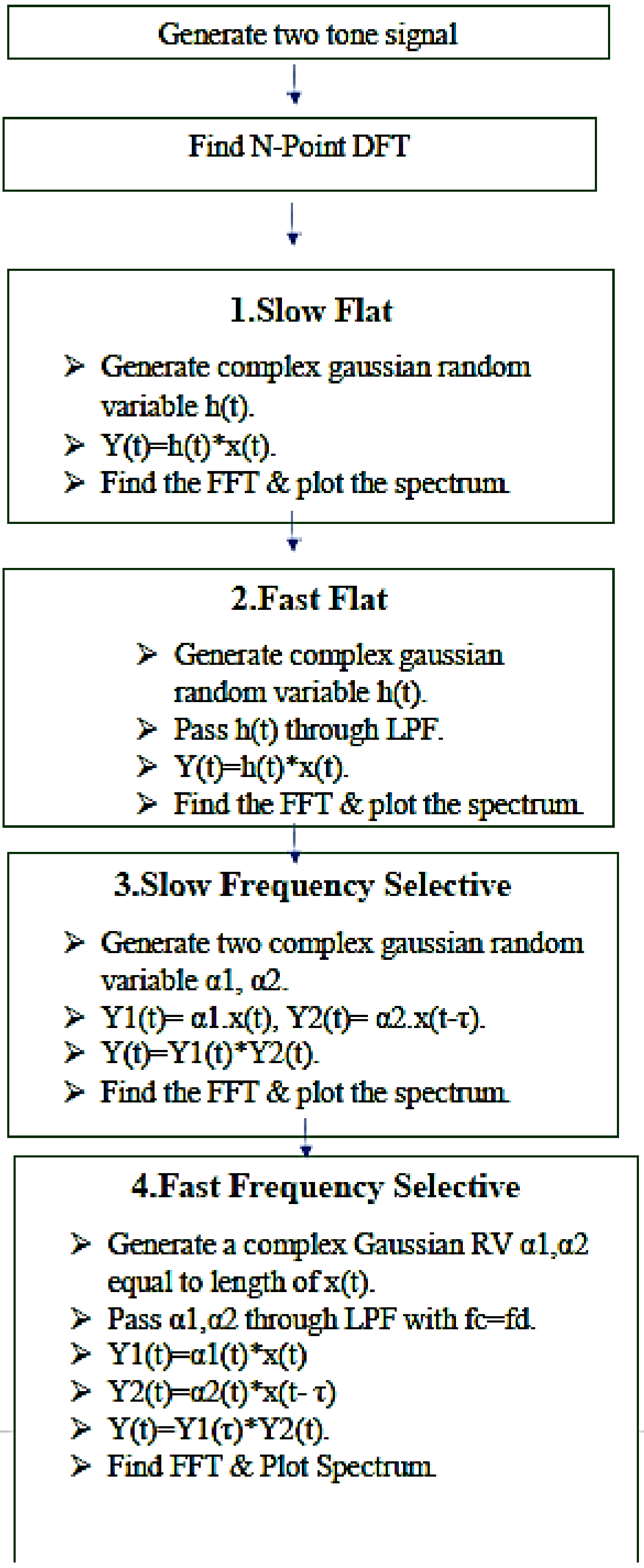
INFERENCE:

Thus in Least square algorithm when iteration increases the error decreases and therefore the mean square error(MSE) decreases.

RESULT:

Thus channel Equalization using least mean square algorithm is carried out successfully and spectrum is spotted. MATLAB code for communication method with zero forcing equalizer was written and performance is analysed

FLOWCHART:



Generate two tone signal

Find N-Point DFT

1.Slow Flat

- Generate complex gaussian random variable h(t).
- Y(t)=h(t)*x(t).
- Find the FFT & plot the spectrum.

2.Fast Flat

- Generate complex gaussian random variable h(t).
- Pass h(t) through LPF.
- Y(t)=h(t)*x(t).
- Find the FFT & plot the spectrum.

3.Slow Frequency Selective

- Generate two complex gaussian random variable α1, α2.
- Y1(t)= α1.x(t), Y2(t)= α2.x(t-τ).
- Y(t)=Y1(t)*Y2(t).
- Find the FFT & plot the spectrum.

4.Fast Frequency Selective

- Generate a complex Gaussian RV α1,α2 equal to length of x(t).
- Pass α1,α2 through LPF with fc=fd.
- Y1(t)=α1(t)*x(t)
- Y2(t)=α2(t)*x(t- τ)
- Y(t)=Y1(τ)*Y2(t).
- Find FFT & Plot Spectrum.

EXP.NO: 8 DATE: 23.4.2022	**Characteristics of wireless channels**

AIM:
To plot the spectra of a two-tone signal due to slow flat fading, slow and frequency selective fading, fast flat fading and flat frequency selective fading.

SOFTWARE REQUIRED:
MATLAB 2019a

THEORY:

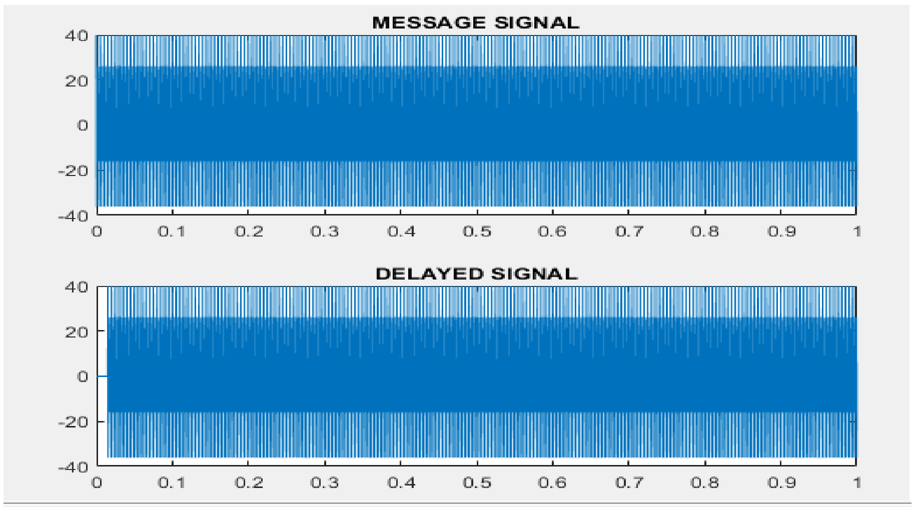
In wireless communications, **fading** is variation of the attenuation of a signal with various variables. These variables include time, geographical position, and radio frequency. Fading is often modelled as a random process. A **fading channel** is a communication channel that experiences fading. In wireless systems, fading may either be due to multipath propagation, referred to as multipath-induced fading, weather (particularly rain), or shadowing from obstacles affecting the wave propagation, sometimes referred to as **shadow fading**.

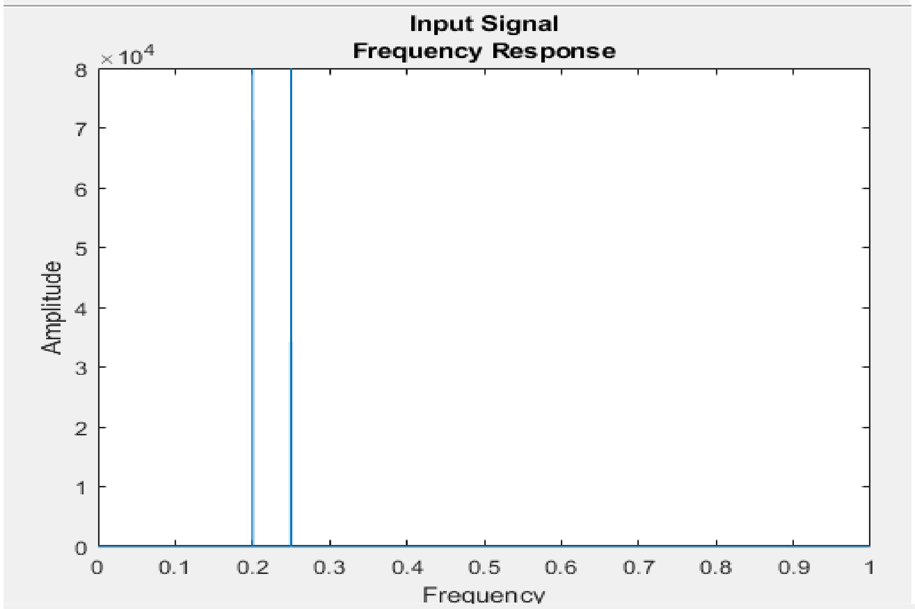
- **Slow fading** arises when the coherence time of the channel is large relative to the delay requirement of the application.[2] In this regime, the amplitude and phase change imposed by the channel can be considered roughly constant over the period of use.

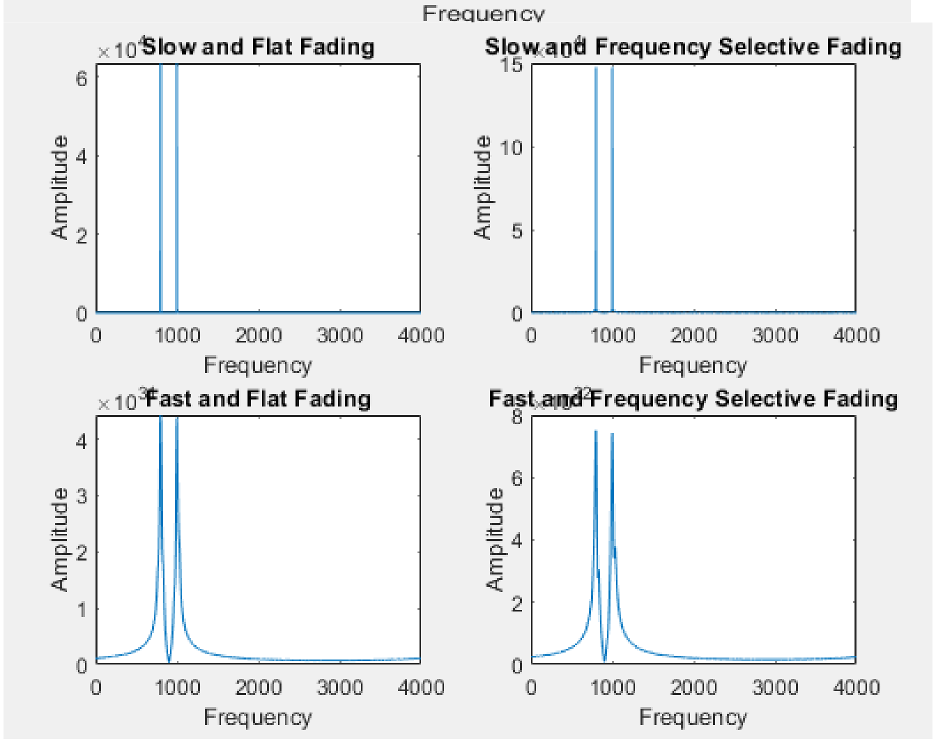
- **Fast fading** occurs when the coherence time of the channel is small relative to the delay requirement of the application. In this case, the amplitude and phase change imposed by the channel varies considerably over the period of use.

- In **flat fading**, the coherence bandwidth of the channel is larger than the bandwidth of the signal. Therefore, all frequency components of the signal will experience the same magnitude of fading.

- In **frequency-selective fading**, the coherence bandwidth of the channel is smaller than the bandwidth of the signal. Different frequency components of the signal therefore experiences uncorrelated fading.


MESSAGE SIGNAL
DELAYED SIGNAL


Input Signal
Frequency Response
Amplitude
Frequency
× 10^4


Slow and Flat Fading
Slow and Frequency Selective Fading
Fast and Flat Fading
Fast and Frequency Selective Fading
Amplitude
Frequency

ALGORITHM:

- Generate a two tone signal.
- Find the N-point DFT of the signal.
- For the slow & fast fading generate the random complex gaussian variable and multiply with the signal.
- Find the FFT of the signal and plot.
- For the Fast & Flat fading generate a gaussian random variable and pass it through LPF & multiply with x(t).
- Find y1(t)+y2(t) and plot.

CODE:

```
clc;
clear all;
close all;
% Sampling Frequency
fs = 4000 ;
% Frequency 1
f1 = 1000;
% Frequency 2
f2 = 800;
ts = 0:(1/fs):1-(1/fs);
signal1 = 20*exp(complex(0,2*pi*f1*ts));
signal2 = 20*exp(complex(0,2*pi*f2*ts));
message = signal1+signal2;
% DELAYING - SPREADING THE SIGNAL
delayed = [zeros(1,60),message(1:length(message)-
60)];
subplot(2,1,1);
plot(ts,message);
title('MESSAGE SIGNAL');
subplot(2,1,2);
plot(ts,delayed);
title('DELAYED SIGNAL');
freqres = fft(message);
figure;
magnitudeplot = abs(freqres);
plot(ts,magnitudeplot);
title({'Input Signal';'Frequency Response'});
xlabel('Frequency');
ylabel('Amplitude');
%Slow and Flat Fading
```

```matlab
h = randn + (i*randn);

y1 = message.*h;
freqres = fft(y1);
figure;
magnitudeplot = abs(fft(y1));
subplot(2,2,1);
plot(1:fs,magnitudeplot(1:fs));

title('Slow and Flat Fading');
xlabel('Frequency');
ylabel('Amplitude');
%Slow and Frequency Selective Fading
h1 = randn + (i*randn);
h2 = randn + (i*randn);
trans2 = (h1.*message) + (h2.*delayed);
magnitudeplot = abs(fft(trans2));
subplot(2,2,2);
plot(1:fs,magnitudeplot);
title('Slow and Frequency Selective Fading');
xlabel('Frequency');
ylabel('Amplitude');
%Fast and Flat Fading
fc = 10;
hs = randn(1,length(ts)) +(i*randn(1,length(ts)));
[b,a] = butter(12,((2*fc)/1000));
lpf = filter(b,a,hs);
trans3 = lpf.*message;
magnitudeplot = abs(fft(trans3));
subplot(2,2,3);
plot(1:fs,magnitudeplot);
title('Fast and Flat Fading');
xlabel('Frequency');
ylabel('Amplitude');
% Fast and Frequency Selective Fading
ht = randn(1,length(ts)) +(i*randn(1,length(ts)));
lpf1 = filter(b,a,ht);
trans4 = (lpf1.*message) +(lpf1.*delayed);
magnitudeplot = abs(fft(trans4));
subplot(2,2,4);
plot(1:fs,magnitudeplot);
title('Fast and Frequency Selective Fading');
xlabel('Frequency');
ylabel('Amplitude');
```

RESULT:

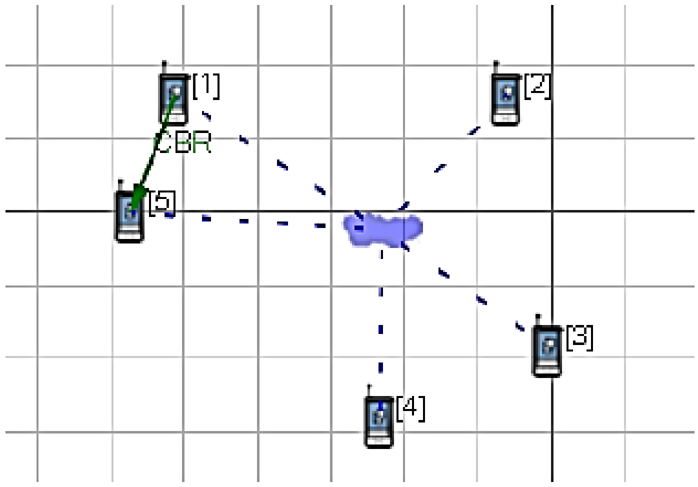
Thus, the four types of fading were simulated and graphs are plotted using MATLAB software.

INFERENCE:

- The amplitude remains the same in slow flat fading.
- The amplitude varies for slow and frequency selective fading.
- The amplitude remains the same but the signal is spread in fast and flat fading.
- The amplitude changes and the signal is spread over the frequencies in fast and frequency selective fading.

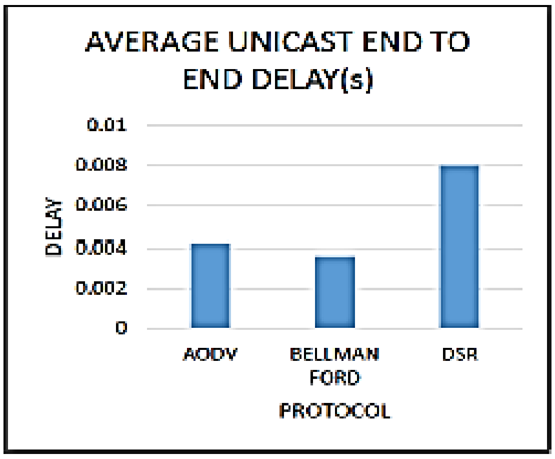
OUTPUTS:

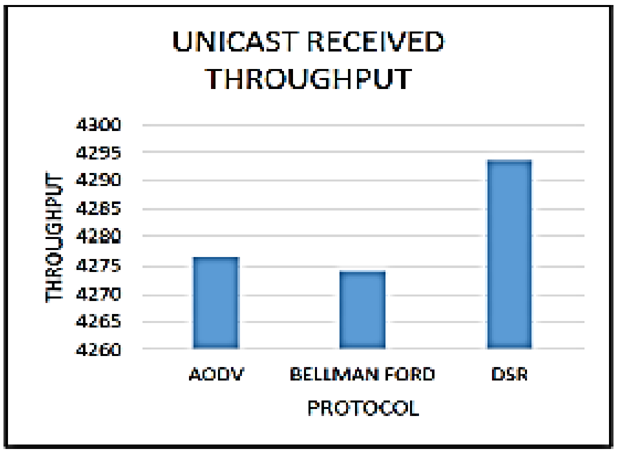
LAYOUT: WITHOUT MOBILITY



TABULATION:

PROTOCOL	AVERAGE UNICAST END TO END DELAY(s)	UNICAST RECEIVED THROUGHPUT
AODV	0.0041981	4276.77
BELLMAN FORD	0.00359559	4274.05
DSR	0.00801132	4293.64





EXP.NO: 9 DATE: 25.4.2022	**Study of wireless protocols using <u>Qualnet</u>**

AIM:

To study and implement wireless routing protocols using qualnet

- AODV-Ad-hoc on demand distance vector
- DSR-Dynamic Static Routing
- Bellman ford

SOFTWARE REQUIRED:

- QUALNET simulator

THEORY:

AODV:AD-HOC ON-DEMAND DISTANCE VECTOR

AODV is a routing protocol for ad-hoc mobile networks with large numbers of mobile nodes.The protocols algorithm creates routes between nodes only when the routes are requested by source nodes giving the network the flexibility to allow nodes to enter and leave the network .Routes remain active only as long as data packets are travelling along the paths from the source to the destination .When the source stops sending packets the path will time out and close.

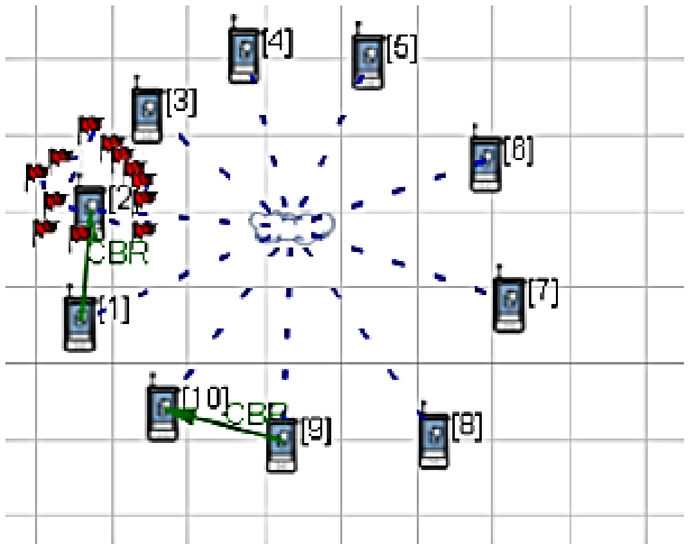
DSR:

Dynamic source routing is a routing protocol for wireless mesh networks. In DSR each source determines the route to be used in transmitting its packets to selected destination. There are two main components called route discovery and route maintenance. Route discovery determines the optimum path for transmission between a given source and destination. Route maintenance ensures that the transmission path remains optimum and loop free as network condition changes, even if this requires changing the route during a transmission.

BELLMAN FORD:

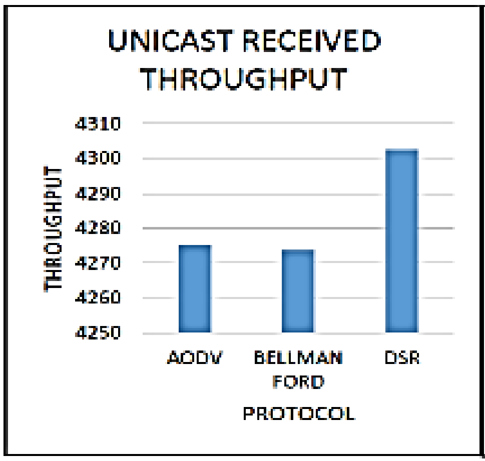
It is used to find the shortest path from one node to all other nodes in a weighed graph. The first step is to initialize the vertices. The algorithm is initially set from the starting vertex to all vertices till infinity. After the initializing step the algorithm starts circulating shortest distance from starting vertex to all other vertices. It is an example of dynamic programming. It follows bottom-up approach.

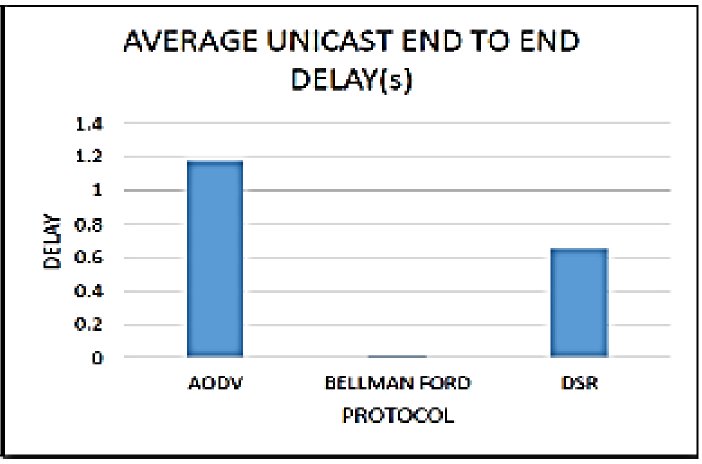
LAYOUT: WITH MOBILITY



TABULATION:

PROTOCOL	AVERAGE UNICAST END TO END DELAY(s)	UNICAST RECEIVED THROUGHPUT
AODV	1.176	4275.19
BELLMAN FORD	0.012	4273.67
DSR	0.656	4302.4





PROCEDURE:

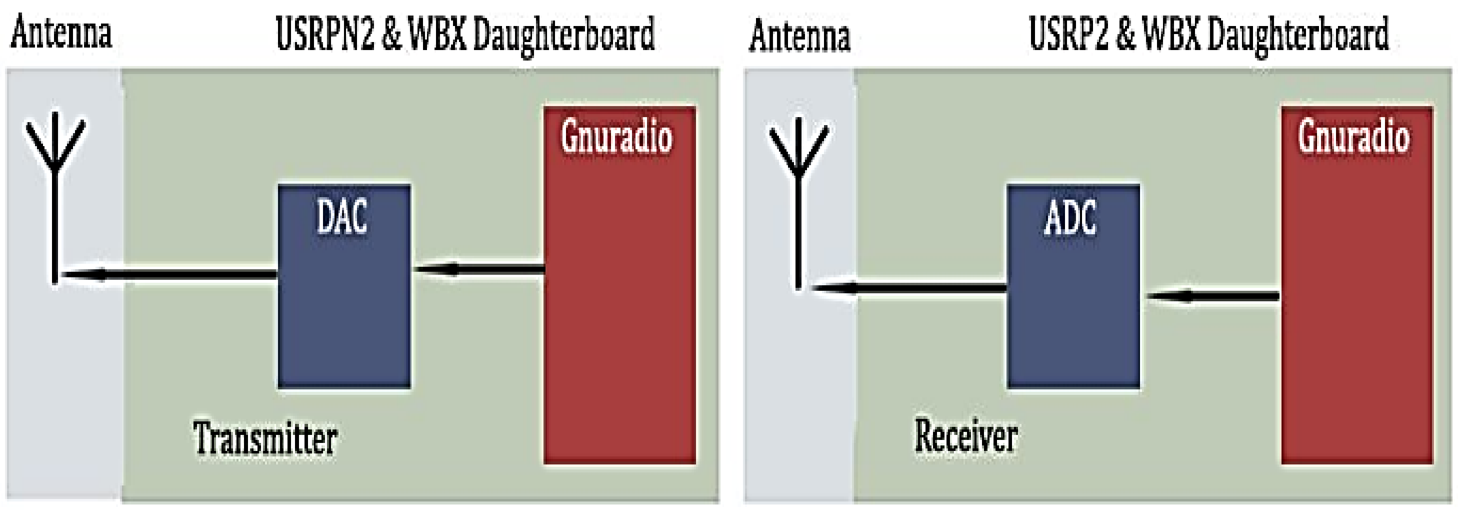
- Open qualnet and set the suitable terrain.
- Place the required nodes and define properties for the nodes and using a subnet.
- In the properties , set the routing protocol to DSR.
- Establish CBR between two nodes.
- Save and run the simulation and there in the analyser window observe the required parameters such as throughput, average end to end delay and average jitter.
- Repeat the steps by varying the node density, (i.e) increase or decrease the number of nodes and tabulate the values.
- The same steps are to be followed for the other routing protocols such as AODV and Bellman Ford .
- The observed metrics are tabulated and plotted for different routing protocols.

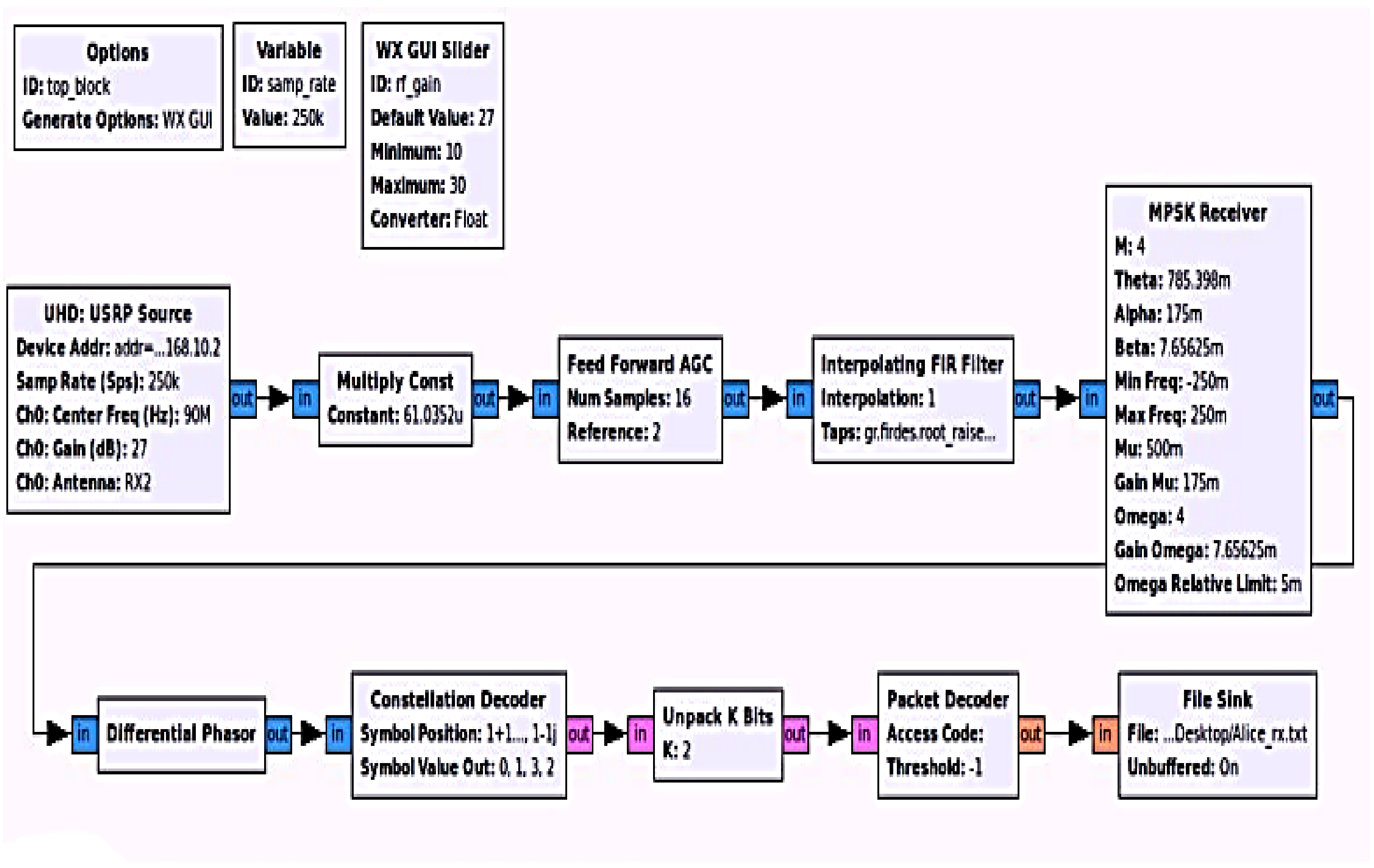
INFERENCE

From the graph, the throughput and delay increase with increase in number of nodes. It is the highest in DSR compared to AODV and Bellman Ford. With mobility as the speed increases, DSR has maximum ed to end delay with AODV has stable delay despite the mobility

RESULT:

The performance evaluation of routing protocols Bellman Ford, AODV and DSR are studied by varying node density using Qualnet.




Options
ID: top_block
Generate Options: WX GUI

Variable
ID: samp_rate
Value: 250k

WX GUI Slider
ID: rf_gain
Default Value: 27
Minimum: 10
Maximum: 30
Converter: Float

UHD: USRP Source
Device Addr: addr=...168.10.2
Samp Rate (Sps): 250k
Ch0: Center Freq (Hz): 90M
Ch0: Gain (dB): 27
Ch0: Antenna: RX2

Multiply Const
Constant: 61.0352u

Feed Forward AGC
Num Samples: 16
Reference: 2

Interpolating FIR Filter
Interpolation: 1
Taps: gr.firdes.root_raise...

MPSK Receiver
M: 4
Theta: 785.398m
Alpha: 175m
Beta: 7.65625m
Min Freq: -250m
Max Freq: 250m
Mu: 500m
Gain Mu: 175m
Omega: 4
Gain Omega: 7.65625m
Omega Relative Limit: 5m

Differential Phasor

Constellation Decoder
Symbol Position: 1+1..., 1-1j
Symbol Value Out: 0, 1, 3, 2

Unpack K Bits
K: 2

Packet Decoder
Access Code:
Threshold: -1

File Sink
File: ...Desktop/Alice_rx.txt
Unbuffered: On

Expanded system in GNU Radio Companion

EXP.NO: 10 **DATE: 25.4.2022**	**Study of SDR based transceiver using USRP and GNU radio**

AIM:
To study about software defined ratio: SDR based transceiver of digital communication system using universal software radio peripheral: USRP family of products

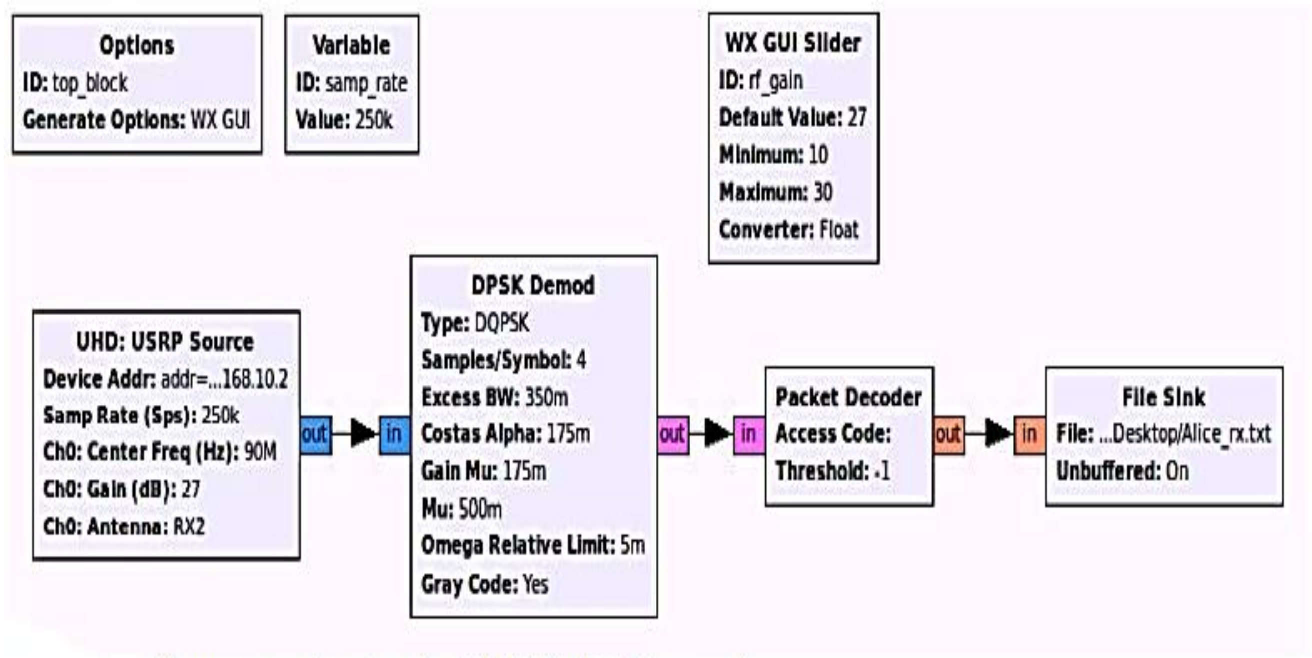
SOFTWARE REQUIRED:
GNU Radio

HARDWARE REQUIRED:
USRP kit

THEORY:

GNU Radio:
GNU Radio is a framework that enables users to design, simulate, and deploy highly capable real-world radio systems. It is a highly modular, "flowgraph"-oriented framework that comes with a comprehensive library of processing blocks that can be readily combined to make complex signal processing applications. GNU Radio has been used for a huge array of real-world radio applications, including audio processing, mobile communications, tracking satellites, radar systems, GSM networks, Digital Radio Mondiale, and much more - all in computer software.
It is, by itself, not a solution to talk to any specific hardware. Nor does it provide out-of the-box applications for specific radio communications standards (e.g., 802.11, ZigBee,LTE, etc.,), but it can be (and has been) used to develop implementations of basically any band-limited communication standard. GNU Radio is a framework dedicated to writing signal processing applications for commodity computers. GNU Radio wraps functionality in easy-to-use reusable blocks, offers excellent scalability, provides an extensive library of standard algorithms, and is heavily optimized for a large variety of common platforms.

USRP kit:
USRP Software Defined Radio device provides a software defined RF architecture to design, prototype and deploy wireless systems with custom signal processing. USRP Software Defined Radio Devices also include options with an onboard FPGA—an SDR game changer that provides wireless communications designers an affordable SDR with unprecedented performance for developing next generation 5G wireless communication systems. USRP Software Defined Radio Devices have an optional onboard GPS that you can use to synchronize multiple software defined radios and enable advanced application possibilities such as distributed multi-channel synchronized systems.



Options
ID: top_block
Generate Options: WX GUI

Variable
ID: samp_rate
Value: 250k

WX GUI Slider
ID: rf_gain
Default Value: 27
Minimum: 10
Maximum: 30
Converter: Float

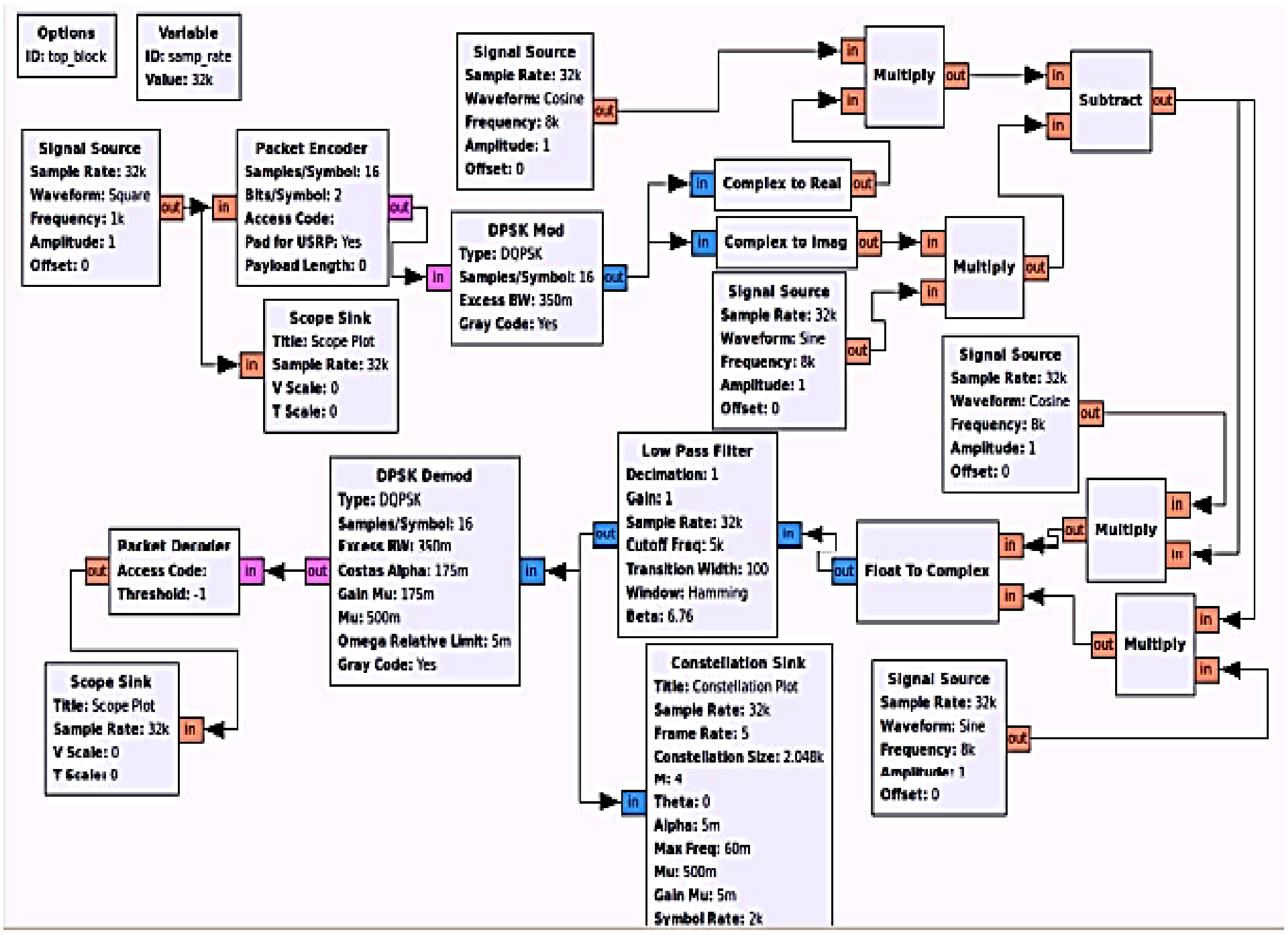
UHD: USRP Source
Device Addr: addr=...168.10.2
Samp Rate (Sps): 250k
Ch0: Center Freq (Hz): 90M
Ch0: Gain (dB): 27
Ch0: Antenna: RX2

DPSK Demod
Type: DQPSK
Samples/Symbol: 4
Excess BW: 350m
Costas Alpha: 175m
Gain Mu: 175m
Mu: 500m
Omega Relative Limit: 5m
Gray Code: Yes

Packet Decoder
Access Code:
Threshold: -1

File Sink
File: ...Desktop/Alice_rx.txt
Unbuffered: On

Compressed system in GNU Radio Companion



Options
ID: top_block

Variable
ID: samp_rate
Value: 32k

Signal Source
Sample Rate: 32k
Waveform: Square
Frequency: 1k
Amplitude: 1
Offset: 0

Packet Encoder
Samples/Symbol: 16
Bits/Symbol: 2
Access Code:
Pad for USRP: Yes
Payload Length: 0

Scope Sink
Title: Scope Plot
Sample Rate: 32k
V Scale: 0
T Scale: 0

Signal Source
Sample Rate: 32k
Waveform: Cosine
Frequency: 8k
Amplitude: 1
Offset: 0

DPSK Mod
Type: DQPSK
Samples/Symbol: 16
Excess BW: 350m
Gray Code: Yes

Multiply

Subtract

Complex to Real

Complex to Imag

Multiply

Signal Source
Sample Rate: 32k
Waveform: Sine
Frequency: 8k
Amplitude: 1
Offset: 0

Signal Source
Sample Rate: 32k
Waveform: Cosine
Frequency: 8k
Amplitude: 1
Offset: 0

Low Pass Filter
Decimation: 1
Gain: 1
Sample Rate: 32k
Cutoff Freq: 5k
Transition Width: 100
Window: Hamming
Beta: 6.76

Multiply

Float To Complex

Multiply

DPSK Demod
Type: DQPSK
Samples/Symbol: 16
Excess BW: 350m
Costas Alpha: 175m
Gain Mu: 175m
Mu: 500m
Omega Relative Limit: 5m
Gray Code: Yes

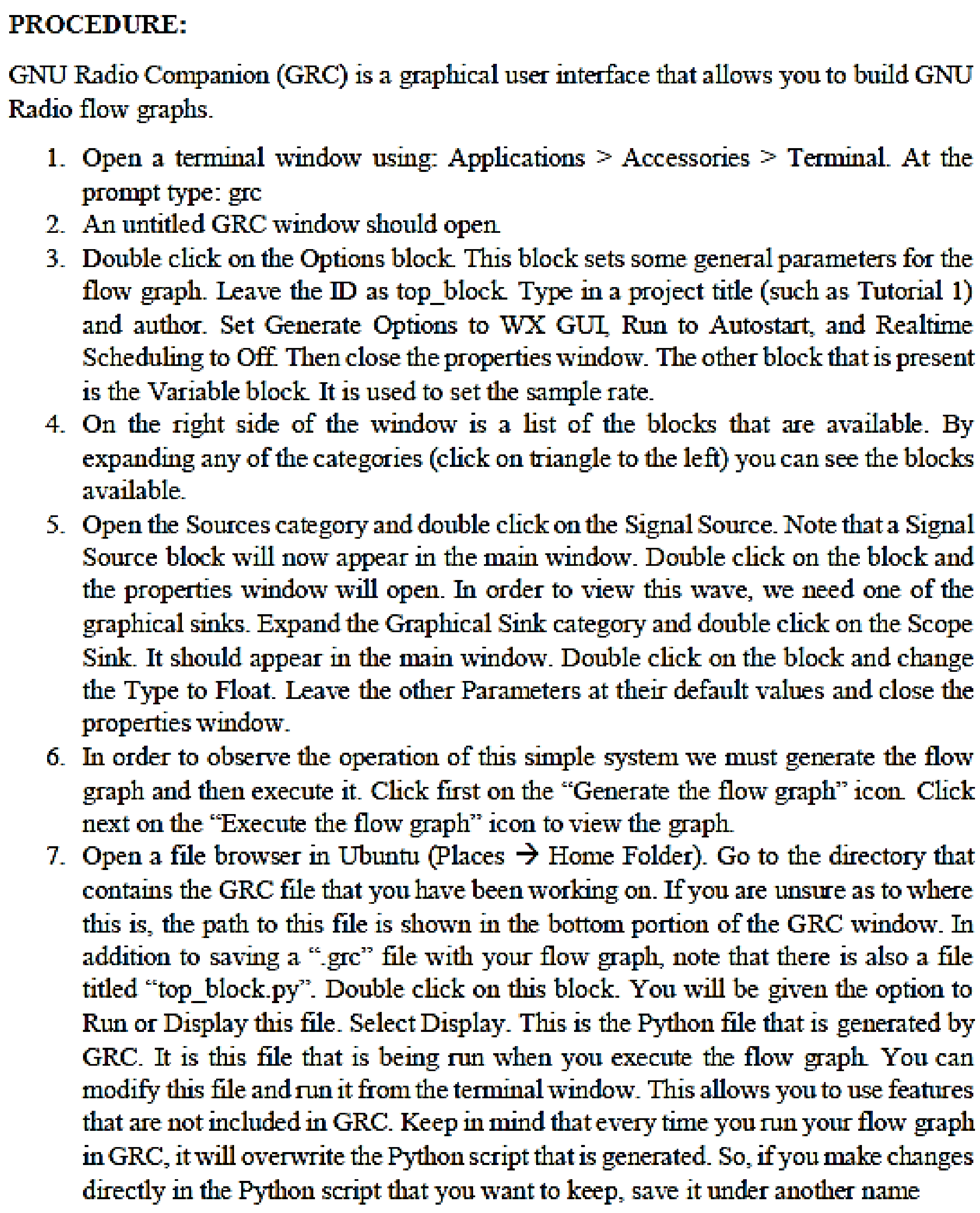
Packet Decoder
Access Code:
Threshold: -1

Scope Sink
Title: Scope Plot
Sample Rate: 32k
V Scale: 0
T Scale: 0

Constellation Sink
Title: Constellation Plot
Sample Rate: 32k
Frame Rate: 5
Constellation Size: 2.048k
M: 4
Theta: 0
Alpha: 5m
Max Freq: 60m
Mu: 500m
Gain Mu: 5m
Symbol Rate: 2k

Signal Source
Sample Rate: 32k
Waveform: Sine
Frequency: 8k
Amplitude: 1
Offset: 0

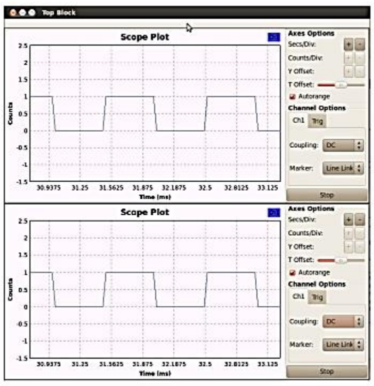
The USRP Software Defined Radio Device is a reconfigurable RF device that includes a combination of host based processors, FPGAs, and RF front ends. The USRP Software Defined Radio Device include options that range from lower cost options with fixed FPGA personalities to high end radios with a large, open FPGAs and wide instantaneous bandwidth. These devices can be used for applications such as multiple input, multiple output (MIMO) and LTE/WiFi testbeds, SIGINT, and radar systems.

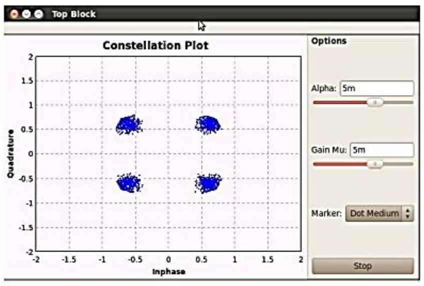


PROCEDURE:

GNU Radio Companion (GRC) is a graphical user interface that allows you to build GNU Radio flow graphs.

1. Open a terminal window using: Applications > Accessories > Terminal. At the prompt type: grc
2. An untitled GRC window should open.
3. Double click on the Options block. This block sets some general parameters for the flow graph. Leave the ID as top_block. Type in a project title (such as Tutorial 1) and author. Set Generate Options to WX GUI, Run to Autostart, and Realtime Scheduling to Off. Then close the properties window. The other block that is present is the Variable block. It is used to set the sample rate.
4. On the right side of the window is a list of the blocks that are available. By expanding any of the categories (click on triangle to the left) you can see the blocks available.
5. Open the Sources category and double click on the Signal Source. Note that a Signal Source block will now appear in the main window. Double click on the block and the properties window will open. In order to view this wave, we need one of the graphical sinks. Expand the Graphical Sink category and double click on the Scope Sink. It should appear in the main window. Double click on the block and change the Type to Float. Leave the other Parameters at their default values and close the properties window.
6. In order to observe the operation of this simple system we must generate the flow graph and then execute it. Click first on the "Generate the flow graph" icon. Click next on the "Execute the flow graph" icon to view the graph.
7. Open a file browser in Ubuntu (Places → Home Folder). Go to the directory that contains the GRC file that you have been working on. If you are unsure as to where this is, the path to this file is shown in the bottom portion of the GRC window. In addition to saving a ".grc" file with your flow graph, note that there is also a file titled "top_block.py". Double click on this block. You will be given the option to Run or Display this file. Select Display. This is the Python file that is generated by GRC. It is this file that is being run when you execute the flow graph. You can modify this file and run it from the terminal window. This allows you to use features that are not included in GRC. Keep in mind that every time you run your flow graph in GRC, it will overwrite the Python script that is generated. So, if you make changes directly in the Python script that you want to keep, save it under another name



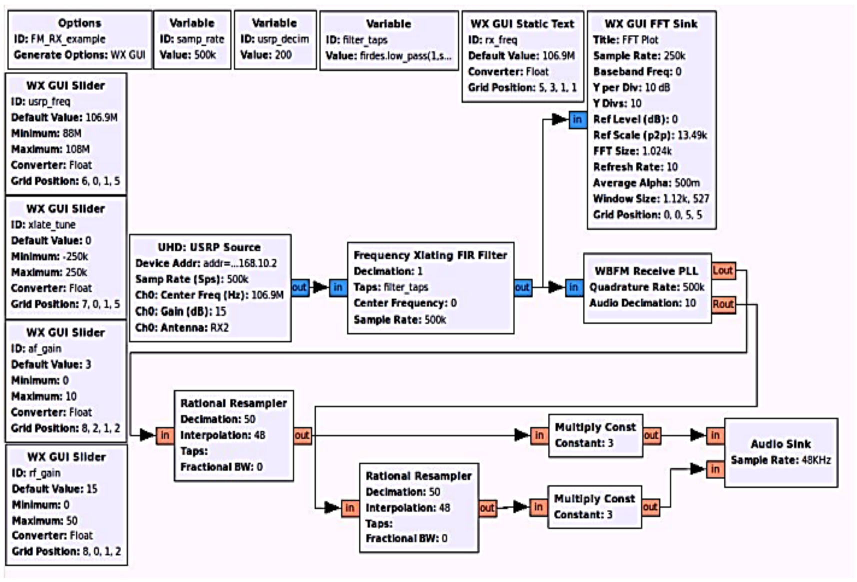


Alpha: 5m

Gain Mu: 5m

Marker: Dot Medium

Left: Transmitted signal and Received signal. Right: The constellation plot of receiver



Options
ID: FM_RX_example
Generate Options: WX GUI

Variable
ID: samp_rate
Value: 500k

Variable
ID: usrp_decim
Value: 200

Variable
ID: filter_taps
Value: firdes.low_pass(1,s...

WX GUI Static Text
ID: rx_freq
Default Value: 106.9M
Converter: Float
Grid Position: 5, 3, 1, 1

WX GUI FFT Sink
Title: FFT Plot
Sample Rate: 250k
Baseband Freq: 0
Y per Div: 10 dB
Y Divs: 10
Ref Level (dB): 0
Ref Scale (p2p): 13.49k
FFT Size: 1.024k
Refresh Rate: 10
Average Alpha: 500m
Window Size: 1.12k, 527
Grid Position: 0, 0, 5, 5

WX GUI Slider
ID: usrp_freq
Default Value: 106.9M
Minimum: 88M
Maximum: 108M
Converter: Float
Grid Position: 6, 0, 1, 5

WX GUI Slider
ID: xlate_tune
Default Value: 0
Minimum: -250k
Maximum: 250k
Converter: Float
Grid Position: 7, 0, 1, 5

WX GUI Slider
ID: af_gain
Default Value: 3
Minimum: 0
Maximum: 10
Converter: Float
Grid Position: 8, 2, 1, 2

WX GUI Slider
ID: rf_gain
Default Value: 15
Minimum: 0
Maximum: 50
Converter: Float
Grid Position: 8, 0, 1, 2

UHD: USRP Source
Device Addr: addr=...168.10.2
Samp Rate (Sps): 500k
Ch0: Center Freq (Hz): 106.9M
Ch0: Gain (dB): 15
Ch0: Antenna: RX2

Frequency Xlating FIR Filter
Decimation: 1
Taps: filter_taps
Center Frequency: 0
Sample Rate: 500k

WBFM Receive PLL
Quadrature Rate: 500k
Audio Decimation: 10

Rational Resampler
Decimation: 50
Interpolation: 48
Taps:
Fractional BW: 0

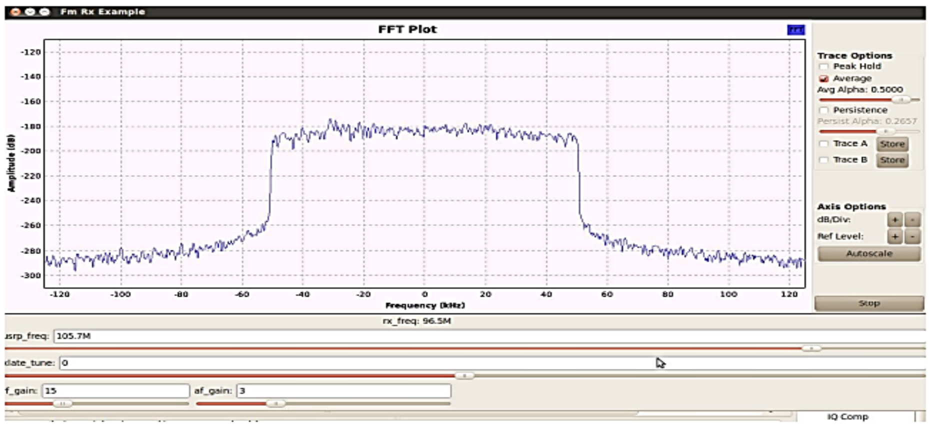
Multiply Const
Constant: 3

Audio Sink
Sample Rate: 48KHz

Rational Resampler
Decimation: 50
Interpolation: 48
Taps:
Fractional BW: 0

Multiply Const
Constant: 3

FM Radio Rx System



Avg Alpha: 0.5000

Persist Alpha: 0.2657

rx_freq: 96.5M

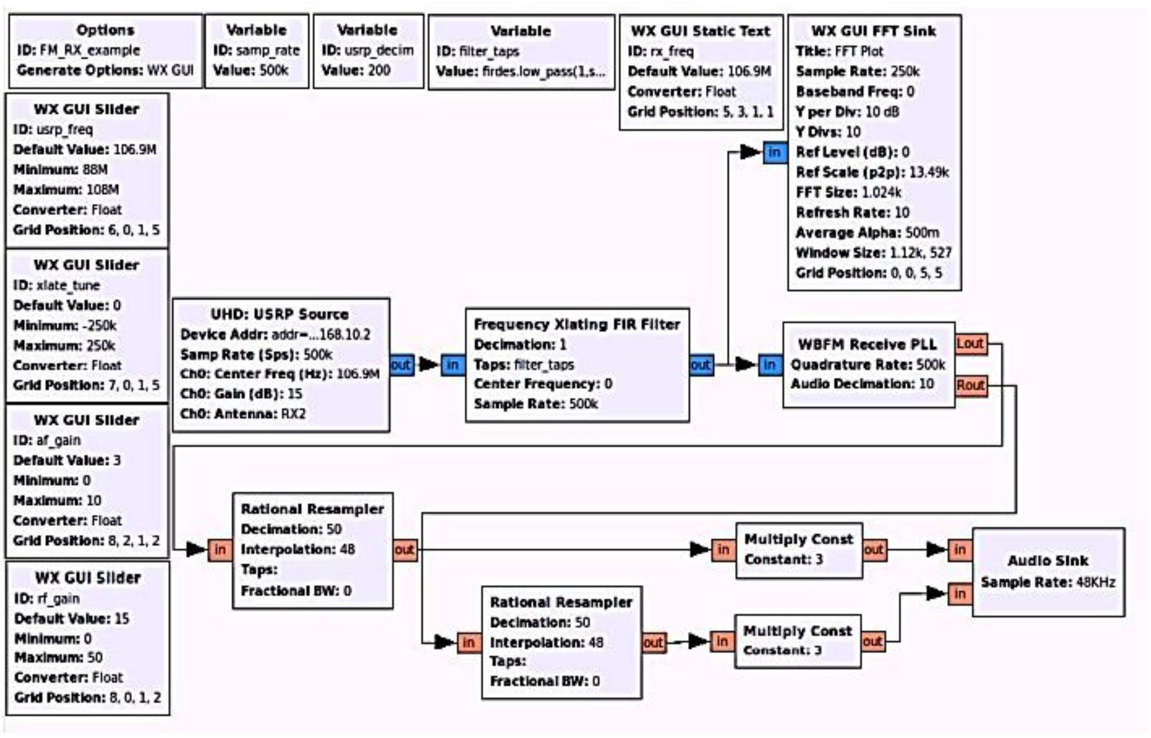
usrp_freq: 105.7M

xlate_tune: 0

rf_gain: 15

af_gain: 3

Received Spectrum of FM Receiver


Options
ID: FM_RX_example
Generate Options: WX GUI

Variable
ID: samp_rate
Value: 500k

Variable
ID: usrp_decim
Value: 200

Variable
ID: filter_taps
Value: firdes.low_pass(1,s...

WX GUI Static Text
ID: rx_freq
Default Value: 106.9M
Converter: Float
Grid Position: 5, 3, 1, 1

WX GUI FFT Sink
Title: FFT Plot
Sample Rate: 250k
Baseband Freq: 0
Y per Div: 10 dB
Y Divs: 10
Ref Level (dB): 0
Ref Scale (p2p): 13.49k
FFT Size: 1.024k
Refresh Rate: 10
Average Alpha: 500m
Window Size: 1.12k, 527
Grid Position: 0, 0, 5, 5

WX GUI Slider
ID: usrp_freq
Default Value: 106.9M
Minimum: 88M
Maximum: 108M
Converter: Float
Grid Position: 6, 0, 1, 5

WX GUI Slider
ID: xlate_tune
Default Value: 0
Minimum: -250k
Maximum: 250k
Converter: Float
Grid Position: 7, 0, 1, 5

WX GUI Slider
ID: af_gain
Default Value: 3
Minimum: 0
Maximum: 10
Converter: Float
Grid Position: 8, 2, 1, 2

WX GUI Slider
ID: rf_gain
Default Value: 15
Minimum: 0
Maximum: 50
Converter: Float
Grid Position: 8, 0, 1, 2

UHD: USRP Source
Device Addr: addr=...168.10.2
Samp Rate (Sps): 500k
Ch0: Center Freq (Hz): 106.9M
Ch0: Gain (dB): 15
Ch0: Antenna: RX2

Frequency Xlating FIR Filter
Decimation: 1
Taps: filter_taps
Center Frequency: 0
Sample Rate: 500k

WBFM Receive PLL
Quadrature Rate: 500k
Audio Decimation: 10

Rational Resampler
Decimation: 50
Interpolation: 48
Taps:
Fractional BW: 0

Rational Resampler
Decimation: 50
Interpolation: 48
Taps:
Fractional BW: 0

Multiply Const
Constant: 3

Multiply Const
Constant: 3

Audio Sink
Sample Rate: 48KHz

- Complete flowgraph with appropriate connections, data types, and parameters

INFERENCE:

A software defined window offers flexibility to deliver the highly reconfigurable system requirements. This approach allows a different type of communication system requirements such as standard, protocol or signal processing method, to be deployed by using the same set of hardware and software such as USRP and GNU radio respectively. However, the realization of SDR concept is inherently limited by analog components of hardware being used.

RESULT and CONCLUSION:

Thus, we have studies the SDR based transceiver system using USRP kit and GNU radio.